University of Stuttgart

Institute for Artificial Intelligence
Intelligent Sensing and Perception

Universitätsstraße 32
70569 Stuttgart

Bachelor Thesis

Domain Adaptation Methods for Emotion and Pain Recognition via Video Games

Jonas Nasimzada

Study program:	Softwaretechnik
1. Examiner:	Jun.-Prof Dr.-Ing Alina Roitberg
2. Examiner:	
Advisors:	Dr.-Ing. Constantin Seibold & Jun.-Prof Dr.-Ing Alina Roitberg
start date:	November 1, 2023
end date:	April 30, 2024

Kurzfassung

Bei der Entwicklung von Systemen zur Interaktion zwischen Patient und Computer ist es von entscheidender Bedeutung, den emotionalen und physischen Zustand des Patienten zu erkennen. Das Sammeln großer Datensätze in sensiblen Situationen wie dem Filmen einer Person mit Schmerzen kann jedoch eine Herausforderung darstellen und ethisch fragwürdig sein.

Das Hauptziel dieser Studie besteht darin, die Möglichkeit der Verwendung synthetischer Daten als alternative Datenquelle zu bewerten, um Modelle zu erstellen, die in der Lage sind, Patientenschmerzen effektiv zu erkennen. Zunächst wurde ein synthetischer Datensatz als Grundlage für die Modellentwicklung erstellt. Um die Relevanz der Vielfalt des synthetisch erzeugten Datensatzes aufrechtzuerhalten, wurde ein 3D-Modell echter Menschen erstellt, indem Gesichtsmerkmale aus einem Quelldatensatz extrahiert und 3D-Netze mit EMOCA (Emotion Driven Monocular Face Capture and Animation) [1] [2] erzeugt wurden. Die Gesichtstexturen wurden aus öffentlich zugänglichen Datensätzen wie CelebHQ [3] und FFHQ-UV [4] entnommen.

Es wurde eine effiziente Pipeline für die Generierung von menschlichen Netzen und Texturen erstellt, die zu einem Datensatz von 8.600 synthetischen menschlichen Modellen führte, die in etwa 2 Stunden pro Perspektive und Textur generiert wurden. Die Datensätze umfassen unterschiedliche Gesichtstexturen und -perspektiven und sind insgesamt über 300 GB groß. Dieser Ansatz verbessert die geschlechtliche und ethnische Vielfalt und bietet gleichzeitig Perspektiven aus bisher unbekannten Blickwinkeln.

Durch die Kombination der 3D-Modelle mit den extrahierten Texturen wurden neue Charaktere mit unterschiedlichen Gesichtstexturen, aber identischen Gesichtsausdrücken geschaffen. Die Studie zielt darauf ab, die Lücke zwischen synthetischen Daten und realen medizinischen Kontexten mit Hilfe von Domänenanpassungsmethoden wie dem Domain Mapping zu schließen. Dieser Ansatz macht menschliche Teilnehmer überflüssig und löst ethische Probleme, die mit traditionellen Datenerfassungsmethoden verbunden sind.

Verschiedene Kombinationen von Datensätzen mit unterschiedlichen Texturen und Perspektiven wurden verwendet, um Modelle zu trainieren und die Anwendbarkeit von synthetischen Daten für die Domänenanpassung (Domain Mapping) mit realen menschlichen Daten als Eingangsvideo zu bewerten.

Die Einbeziehung von synthetischen und realen Daten führt zu einer verbesserten Schmerzerkennung. Der kombinierte Ansatz ermöglicht die Nutzung der Stärken von realen und synthetischen Datensätzen, was zu einem robusteren und effektiveren Modell für die Schmerzerkennung führt.

Abstract

Seeing the patient's emotional and physical condition is crucial when designing patient-computer interaction systems. However, gathering large datasets in sensitive situations like filming a person in pain can be challenging and ethically questionable.

The primary aim of this study is to assess the possibility of using synthetic data as an alternative data source to create models capable of effectively recognizing patient pain. Initially, a synthetic dataset was generated as the foundation for model development. To maintain the relevance of the synthetically generated dataset's diversity, a 3D model of real people was created by extracting facial landmarks from a source dataset and generating 3D meshes using EMOCA (Emotion Driven Monocular Face Capture and Animation) [1] [2]. Meanwhile, facial textures were sourced from publicly available datasets like CelebHQ [3] and FFHQ-UV [4].

An efficient pipeline was created for human mesh and texture generation, resulting in a dataset of 8,600 synthetic human heads generated in approximately 2 hours per perspective and texture. The datasets encompass varying facial textures and perspectives and total over 300 GB. This approach enhances gender and ethnic diversity while introducing perspectives from previously unseen viewpoints.

Combining the 3D models with the extracted textures created new characters with varying facial textures but identical facial expressions. The study aims to bridge the gap between synthetic data and real-world medical contexts using domain adaptation methods, like Domain Mapping. This approach eliminates the need for human participants and addresses ethical issues associated with traditional data collection methods.

Different combinations of datasets, encompassing various textures and perspectives, were utilized to train models and assess the feasibility of synthetic data for domain adaptation (Domain Mapping) with real human data as input video.

However, incorporating synthetic and real data leads to improved pain recognition capabilities. This combined approach can leverage the strengths of both real and synthetic datasets, resulting in a more robust and effective model for pain recognition.

Contents

List of Figures

List of Tables

Acronyms

2D Two-dimensional.

3D Three-dimensional.

AI Artificial Intelligence.

AU Action Units.

AUROC Area Under the Receiver Operating Characteristic Curve.

CNN Convolutional Neural Network.

EMOCA Emotion Driven Monocular Face Capture and Animation.

FACS Facial Action Coding System.

FFHQ Flickr-Faces-HQ.

FFHQ-UV Normalized Facial UV-Texture Dataset for 3D Face Reconstruction.

FLAME Faces Learned with an Articulated Model and Expressions.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

GAN Generative Adversarial Networks.

IASP International Association for the Study of Pain.

ML Machine Learning.

PSPI Prkachin and Solomon Pain Intensity.

ReLU Rectified Linear Unit.

ROC Receiver Operating Characteristic.

SGD Stochastic gradient descent.

TN True Negative.

TP True Positive.

TPR True Positive Rate.

1 Introduction

Chronic pain, a relentless adversary impacting millions globally, presents a significant burden on healthcare systems. Traditionally, pain assessment relies heavily on subjective self-reporting, a method susceptible to cultural biases, emotional states, and individual interpretation. While physiological signals like heart rate offer an alternative, they can be influenced by factors like anxiety or physical exertion. In the quest for a more objective and reliable approach, automatic pain assessment using facial expressions emerges as a compelling alternative.

Human facial expressions, a rich tapestry of emotions, can reveal valuable clues about a person's internal state, including pain. By analyzing these intricate movements and features, automatic pain recognition systems hold the potential to revolutionize pain assessment. Imagine a future where a simple camera, coupled with sophisticated algorithms, could provide an objective measure of pain intensity, aiding healthcare professionals in diagnosis, treatment planning, and pain management.

However, collecting real-world pain data to train and validate these systems faces significant challenges. Ethical considerations, participant recruitment, complex data collection procedures, the need for subjective pain labeling, and practical limitations in capturing spontaneous pain episodes all contribute to the high cost, time consumption, and limited volume of available real-world data.

Collecting real-world pain data is often costly and time-consuming, with data volume limited by practical constraints. Human subject studies need ethical approval from IRBs to ensure participant safety. This lengthy process involves protocol revisions for ethical compliance [5].

Recruiting participants can be challenging, as it requires targeting specific patient populations with diagnosed chronic pain, potentially limiting the generalizability of findings [6]. Data collection may involve multiple steps, such as using high-resolution cameras, controlled environments, and trained personnel to gather reliable data. Participants may also provide self-reported pain ratings alongside video recordings of facial expressions [6].

Data labeling for machine learning involves assigning pain intensity to data, a subjective and potentially expensive process [7]. Practical limitations include collecting data in controlled settings that may not represent natural pain variations. Capturing spontaneous pain episodes in real-world situations can be even more difficult and costly, often requiring researchers to follow participants in their natural environments or use wearable recording devices [6].

Synthetic-generated video data offers a potential solution to the limitations associated with real-world pain data. Through computer graphics techniques, synthetic videos can be meticulously generated, creating vast and diverse datasets with meticulously controlled variations. This level of control and diversity can significantly improve the generalizability of pain recognition models, enabling them to perform more robustly in real-world scenarios.

This study created an efficient pipeline for human mesh and texture generation, resulting in a dataset of 8,600 synthetic human heads generated in approximately 2 hours per perspective and texture. The datasets encompass varying facial textures and perspectives and total over 300 GB. This approach enhances gender and ethnic diversity while introducing perspectives from previously unseen viewpoints.

The model was trained using different combinations of datasets, textures, and perspectives to assess the feasibility of synthetic data in domain adaptation (Domain Mapping) for real human data as input video.

This work's structure starts with exploring the theoretical background regarding facial expressions during pain, providing foundational knowledge on the subject. This is followed by an introduction to machine learning and its evaluation metrics, offering insights into the approaches and methods used in the study. Subsequently, 3D polygon mesh and UV texture are introduced to provide context for the technological aspects.

Next, the related works are presented, reviewing existing research and developments in the field. This leads into a chapter on dataset generation, outlining how the data was created for this work, and then the deep learning model design is detailed.

The experiment setup chapter discusses the various test cases and the conditions under which the experiments were conducted. Finally, the results of the study are presented and analyzed, culminating in the conclusion and outlook, which summarize the findings and propose directions for future research.

2 Theoretical Background

This chapter presents an overview of the theoretical concepts and methodologies that form the basis for the research conducted in this thesis. These topics provide a comprehensive foundation for understanding the challenges and opportunities in developing machine learning models for pain recognition using facial expressions.

2.1 Analyzing Facial Expressions During Pain

According to the International Association for the Study of Pain (IASP), pain is defined as ”an unpleasant sensory and emotional experience associated with actual or potential tissue damage or described in terms of such damage” [8]. It is considered an indicator of the current health situation and identifies harmful conditions for the body [9].

Pain is a subjective experience, and self-report is considered the most widely used measure of suffering [10]. However, relying solely on self-reporting may not always be reliable or valid, especially for demented patients. Additionally, it cannot be used for unconscious or newborn patients [11].

One potential solution is to utilize facial expressions. The human face provides a wealth of non-verbal information about an individual's health condition [12]. Facial expressions can be viewed as reflective and spontaneous reactions to painful experiences [13]. The following section will discuss Measuring Expressiveness.

2.1.1 Measuring Expressiveness

The Facial Action Coding System (FACS) is a standardized system used to identify, describe, and measure the expressiveness of human faces based on the underlying musculature. Contraction of facial musculature moves parts of the skin, causing visible changes in the face. These changes are the cues for the identification of a particular movement [14].

Facial movements produced by mimetic muscles are called Action Units (AU). Each action unit is assigned a numerical code and a descriptive name. For instance, the code 'AU1' refers to 'Action Unit 1 - Inner Brow Raiser'. This code is used when the inner part of the eyebrows is raised due to the medial part of the frontalis muscle on the human forehead contracting [15].

The Facial Action Coding System (FACS) allows for the systematic and objective identification of facial movements based solely on the visible changes in the appearance of the face resulting from the underlying muscle contractions [14]. Figure 2.1 provides an illustrative example of a facial pain expression classified according to the FACS in terms of seven distinct facial components.

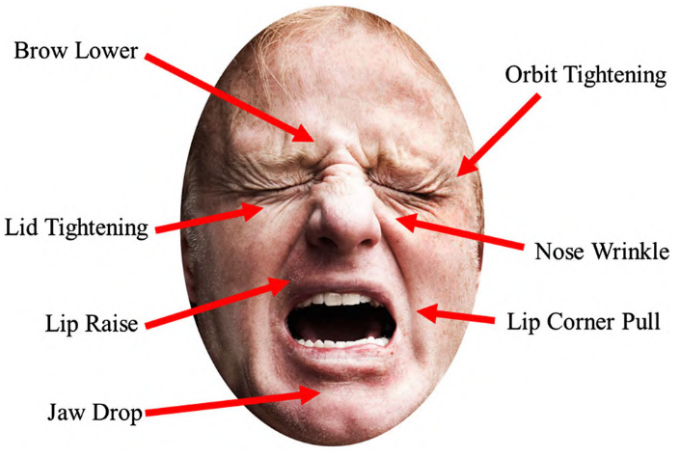


Figure 2.1: An example of facial action coding [16]

Figure 2.2 illustrates the label example images for a specific set of AUs. In later versions, the criteria have changed so that, for Example, AU25, AU26, and AU27 for the upper face AU and AU41, AU42, and AU43 for the lower face AU are merged according to intensity criteria [17].

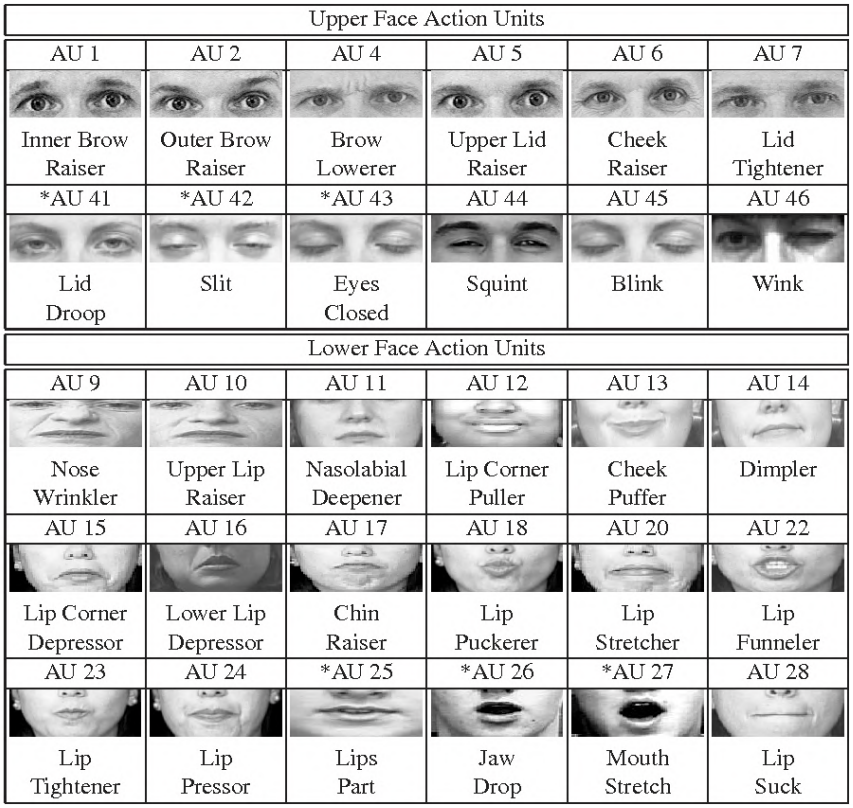


Figure 2.2: FACS action units [18]. *Indicate that the criteria have changed for this AU in later versions [17]

AU may occur individually or in combination. Multiple AUs can appear simultaneously. These combinations can be divided into additive and nonadditive categories. While the AUs with additive combinations retain their original characteristics, nonadditive combinations can cause changes to the characteristics of the single AU [17].

An example of an additive combination is AU 1+2, as seen in Figure 2.3. For nonadditive combinations, AU 1+4 is an illustrative example [19]. Approximately 7,000 AU combinations are currently observed, which presents a more significant challenge for automatic recognition of AUs [20].

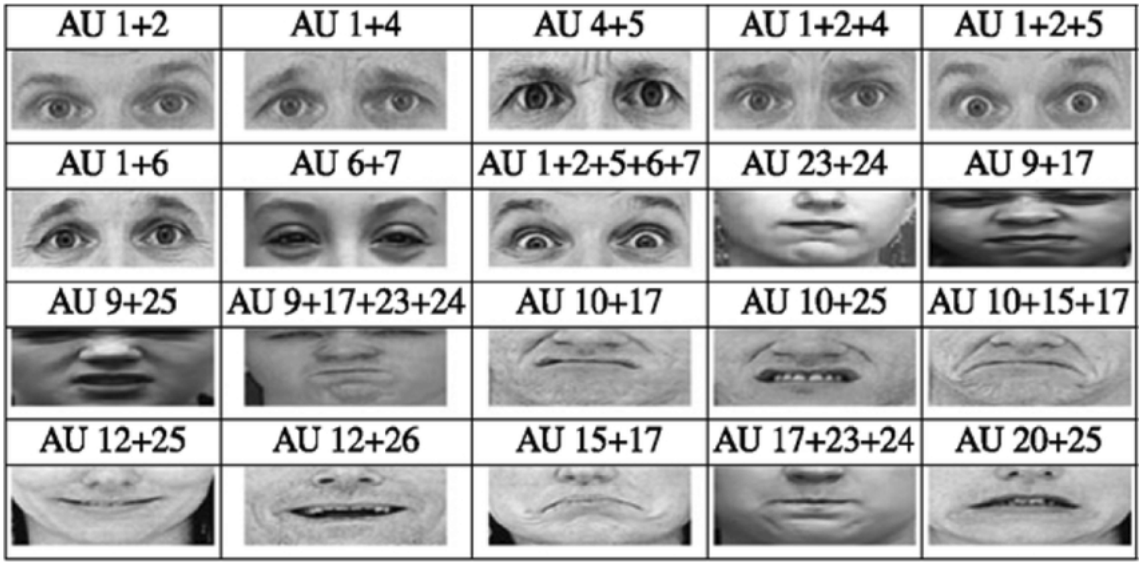


Figure 2.3: Examples of combination of facial action units [18].

The expression of pain is coded by a set of Action Units (AUs) and their corresponding activation of a small set of facial muscles [21]:

- AU 4: Brow lowering
- AU 6 + AU 7: Orbital tightening
- AU 9 + AU 10: Levator labii raise
- AU 43: Eye closure

AU 43 is measured on a binary scale, while the remaining AUs are measured on a six-point ordinal scale (0 = absent, 5 = maximum).

The Prkachin and Solomon Pain Intensity (PSPI) was introduced based on the FACS system. To validate the PSPI score for videos, the maximum PSPI per video is used, as PSPI is calculated per frame. Higher PSPI scores correspond to facial expressions that indicate greater pain intensity, while lower PSPI scores correspond to facial expressions that indicate less pain intensity or possibly no pain [22].

The formula for the PSPI score is defined as:

(2.1)

Pain = Intensity (AU4) + (Max Intensity of AU6 or AU7) + (Max Intensity of AU9 or AU10) + Intensity (AU43)
= AU4 + (AU6||AU7) + (AU9||AU10) + AU43 [22]

Figure 2.4a illustrates a face of pain with the corresponding AUs and their intensities. The PSPI score, calculated according to the formula 2.1, is 12 out of a possible 16. Figure 2.4b compares the PSPI scores [21]. Due to the infrequency of some pain gradations, the PSPI values are typically presented as a summary on a four- to six-point scale [23] [24].

(a) Example of a painful face with the corresponding Action Units and their intensities [21]. (i= intensity of each AU)

(b) Examples with the corresponding pain intensity using the PSPI metric [21]

Figure 2.4: Examples from the UNBC- McMaster Shoulder Pain Expression Archive [23]

The maximum calculated PSPI is used to sort the individual videos of the BioVid Heat Pain Database [24] into the individual classes (BLN, PA1-PA4) (see chapter 4.1).

2.2 Introduction to Machine Learning

Machine learning (ML), a branch of artificial intelligence, empowers systems to gain insights from data and execute tasks through explicit programming. By leveraging diverse algorithms, it absorbs information to refine its understanding, interprets patterns within datasets, and forecasts future outcomes. Through extensive data training, these algorithms yield models that drive task-oriented outputs in ML [25].

As Tom Mitche stated, "The computer program is said to learn from experience E concerning some class of tasks T and performance measure P if its performance at tasks in T."[26]. According to this quote, the computer acquires knowledge from data (E), autonomously executes tasks, and renders decisions (T), and its effectiveness (P) can be assessed through evaluation.

ML broadly serves two main functions. First, it solves tasks that are too complex for humans to perform. A major strength of these programs is their ability to quickly analyze large and complex datasets, enabling feats such as weather forecasting, web search engine optimization, and astronomical data analysis. The ability to identify patterns within large datasets is an up-and-coming area, as these programs can learn with almost unlimited memory, aided by the continued acceleration of processor speeds [25].

Second, ML promotes adaptability. Unlike conventionally programmed tools, which remain static once created, program learning introduces flexibility and adaptability. Based on input, they dynamically adjust to changes in their environment and offer solutions accordingly. Typical applications include systems that need to adapt to changing user behavior, such as spam detection, speech recognition, or, like in this work, the detection of pain in humans [27].

2.2.1 Machine learning techniques

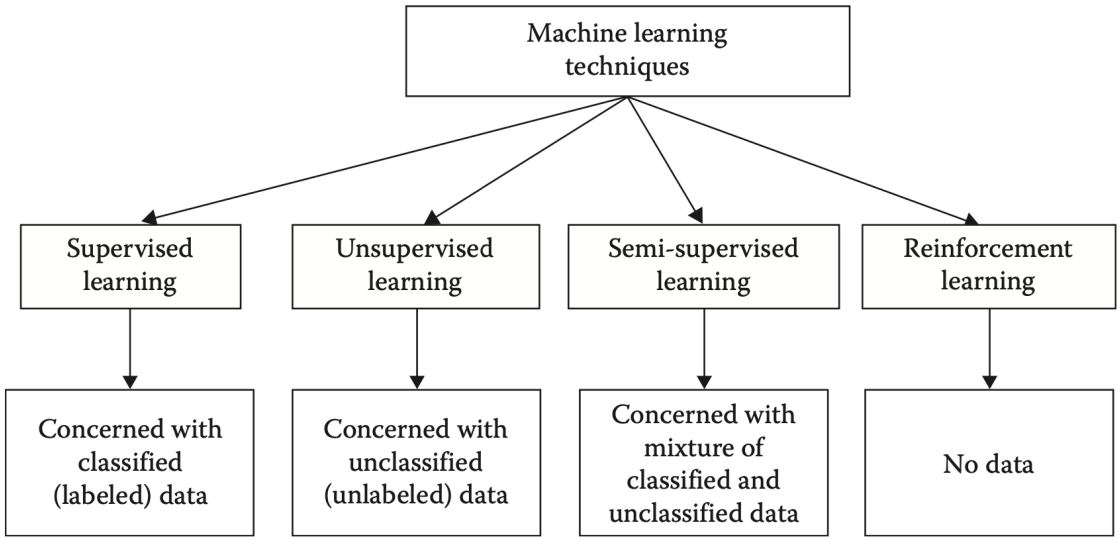


Figure 2.5: Different machine learning techniques and their required data [25]

Figure 2.5 illustrates four techniques and briefly describes the data type needed. Supervised learning involves training a model to map inputs to outputs based on data that includes labels. The training dataset consists of an input vector X paired with an output vector Y, where each element of Y serves as the label explaining the corresponding example from X, forming a complete training example [28]. The output vector Y contains the training data's labels for each training example. The field of supervised learning algorithms encompasses two primary categories: regression and classification [25]. In the context of regression, algorithms are designed to forecast the probability of an outcome in each subject, frequently extending beyond the data upon which they are based [29]. This extension is often referred to as extrapolation. In the case of classification, algorithms are employed to classify data into predefined categories [30]. This study will use a supervised learning model to classify binary whether a human experiences pain or not.

Unsupervised learning is an algorithm that learns from plain examples without any associated input, leaving it up to the algorithm to determine the patterns in the data on its own. As a subset of machine learning, it is similar to the methods humans use to determine that particular objects or events belong to the same class, for example, by observing the degree of similarity between objects [31].

Semi-supervised learning combines classified and unclassified information to generate a suitable data classification model. Typically, labeled datasets are limited, and unlabeled datasets are abundant. Classification models are designed to be more accurate in predicting classes of future test data than models based on only labeled data [25].

Like unsupervised learning, reinforcement learning provides the algorithm with unlabeled examples. The difference is that reinforcement learning offers positive or negative feedback to the algorithm. This type of learning is beneficial for decision-making applications where the algorithm's decisions have consequences. It can be compared to learning by trial and error in the human world. Mistakes aid learning by imposing a penalty, indicating that a particular approach is less likely to succeed than others [31].

2.2.2 Deep Learning and Neural Networks

Deep learning is a subset of machine learning that emphasizes the iterative learning of representational layers from data. The defining characteristic of deep learning is the depth of a model, which is determined by the number of layers. Nowadays, deep learning usually consists of many layers that automatically learn models from training data. These layers are known as neural networks and are structured as literal layers stacked on each other. Neural nets usually consist of three or more layers: Input, Hidden, and Output [31].

The input layer is the initial recipient of data, signals, features, or measures from the external environment as a form of information. Typically, these inputs are normalized within the limits generated by activation functions, also known as samples or patterns. This normalization improves the numerical precision of the mathematical operations performed by the network [32].

The hidden layers are responsible for extracting patterns associated with the analyzed process or system. Most of the internal processing is done in these layers. The output layer presents the final network outputs resulting from the previous layers' processing [33]. Inspired by the organization of neurons in the human brain, the concept of neural networks is inspired by neurobiology [34].

Through training, these networks allow computers to solve abstract and ill-defined problems by mimicking how the brain works. Deep learning models typically comprise tens of thousands or even millions of processing nodes, which are interconnected in a highly nested manner. In contrast to conventional neural networks, deep learning models often include multiple hidden layers, which are employed to address increasingly complex problems [31].

2.2.3 Structure of Neural Networks

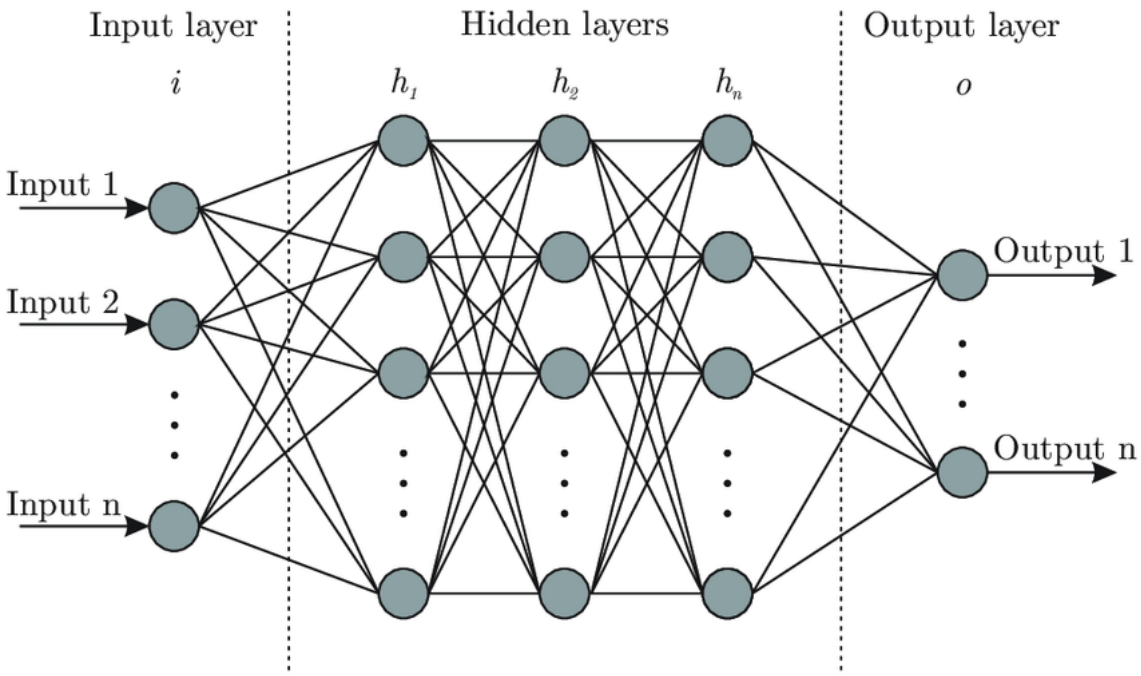


Figure 2.6: Artificial neural network [35]

Figure 2.6 shows the detailed structure of an artificial neural network, which is a type of neural network. Neurons, represented by circles in figure 2.6, are the fundamental elements of it. They receive and transmit information to other cells in the layers [36]. The neurons receive input values $x = [x_1, x_2, ..., x_n]$, which are each multiplied by a specific weight $w = [w_1, w_2, ..., w_n]$ The logits of the neurons, defined as

$$z = \sum_{(i=0)}^{n} w_i x_i \tag{2.2}$$

are obtained by summing the weighted inputs. The activation function indicates which neurons are active. The output is $y = f(z + b)$, where b is the bias term. The output y may be directed to another neuron [36], and various activation functions are available, including Sigmoid, Tanh, ReLU, and Softmax [36].

2.2.4 Training Process of a Neural Network

The training process for a neural network involves an iterative adjustment of its parameters, which consist of weights and biases (represented by an arrow), to minimize a loss function that measures the error between the model's predictions and the actual target values [37]. The learning process can be divided into four main parts: forward propagation, loss calculation, backward propagation (backpropagation), and parameter updates, as seen in Figure 2.7.

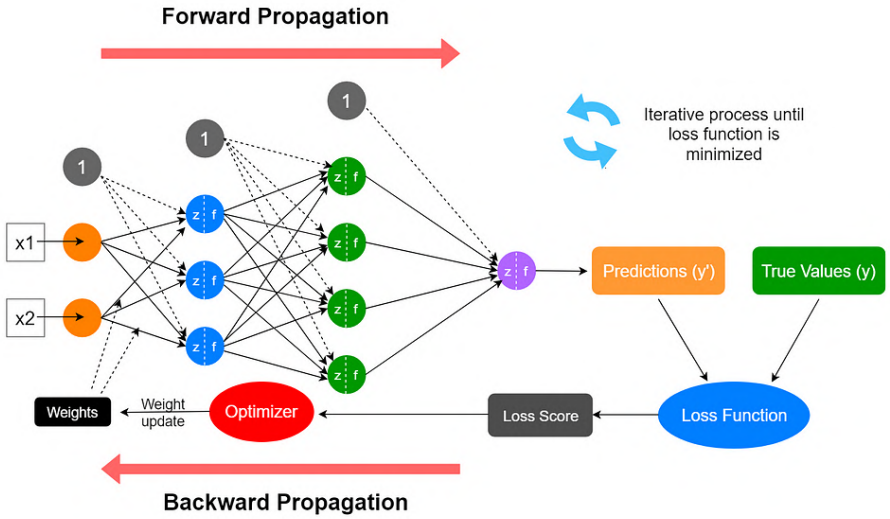


Figure 2.7: Training Cycle of a Neural Network [38]

1. **Forward Propagation:**
   Forward propagation, also known as the forward pass, passes data through the network's layers to generate predictions. The input data is fed into the network, and each layer applies a transformation using weights and biases to produce an output. These outputs then serve as inputs for the next layer, and this process continues until the final output layer is reached [39].

   The process begins with input data, which can be any kind of data, such as images, text, or numerical data. The network's parameters are applied to the input data as it moves through each layer. The weights control the importance of each input, while biases determine how easily a neuron fires or activates. Neurons apply activation functions to the input data, allowing the network to capture non-linear relationships. At the end of forward propagation, the network produces an output prediction compared to the valid target values to calculate the loss [39].

2. **Calculation of the Loss Function:**
   Once the predictions are made, the loss function calculates the error between the predictions and the valid target values. This loss function produces a single value representing how well the model's predictions match the valid values. A higher loss indicates more error in the model's predictions, signaling a need for parameter updates [37].

3. **Backpropagation:**
   Backpropagation is a gradient descent method that calculates the gradients of the loss function concerning each parameter in the network. These gradients are then used to update the network's parameters to minimize the loss function. The process starts at the output layer and moves backward through the network, calculating gradients at each layer (Error Propagation). Each Gradient is computed using the calculus chain rule, determining how much each parameter needs to change to reduce the loss [40].

4. **Parameter Updates:**
   Once the gradients have been calculated, the model's parameters are updated using an optimization algorithm. Standard optimizers include Stochastic Gradient Descent (SGD) and Adam. The SGD updates parameters based on a single data point or a batch of data points, making iterative adjustments to the model. In contrast, the Adam is an adaptive optimizer that adjusts the learning rate based on the average and variance of past gradients, allowing for efficient and stable training [37].

   The optimization algorithm also uses a hyperparameter named learning rate, which determines the size of the parameter updates. A small learning rate results in slower, more stable training, while a more extensive learning rate can speed up training but may lead to instability. From the steps below, the optimizer uses the gradients and the learning rate to adjust the model's parameters to minimize the loss [37].

This iterative process of forward propagation, loss calculation, backpropagation, and parameter updates continues until the model reaches an acceptable level of performance.

2.2.5 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are specialized neural networks designed to handle data arranged in a grid structure, such as time series data or images. Mathematically, convolution involves combining two functions to modify one function based on the shape of the other. CNNs use this operation to derive features from raw pixel data inputs, enabling them to interpret and make predictions about objects within an image [41][42].

As shown in Figure 2.8, the process begins with raw data, such as images, and utilizes convolutional layers to extract progressively more complex features. After the convolutional layers extract features, the data is passed through fully connected layers. These layers perform tasks such as classification or regression by taking the extracted features and making predictions based on them. Finally, the network produces its prediction based on the processed information [42].

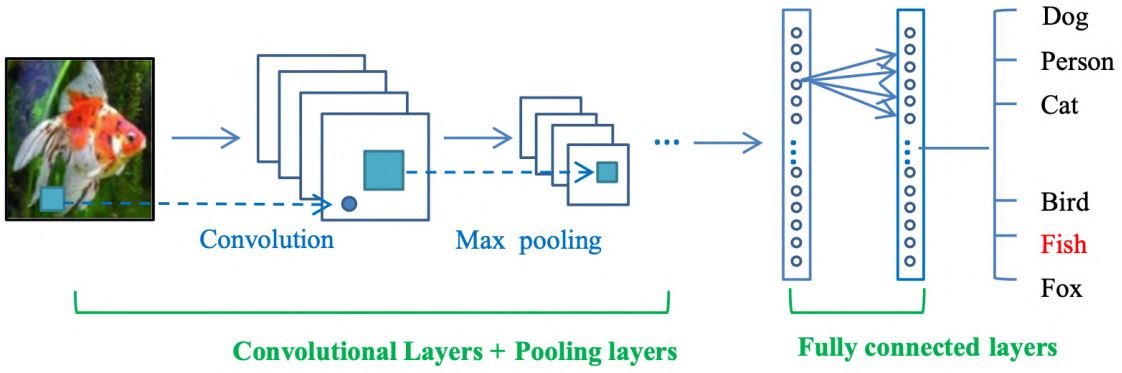


Figure 2.8: The pipeline of the general CNN architecture [43]

CNNs designed to classify typically consist of layers organized by functionality, with each layer feeding into the next. An input feature map generated from the raw pixel data of an image is fed into the CNN. In developing a feature map, a three-dimensional matrix is created from an image's raw pixel data using the CNN. The first two dimensions represent the image's width and length, while the third dimension corresponds to the three color channels [42].

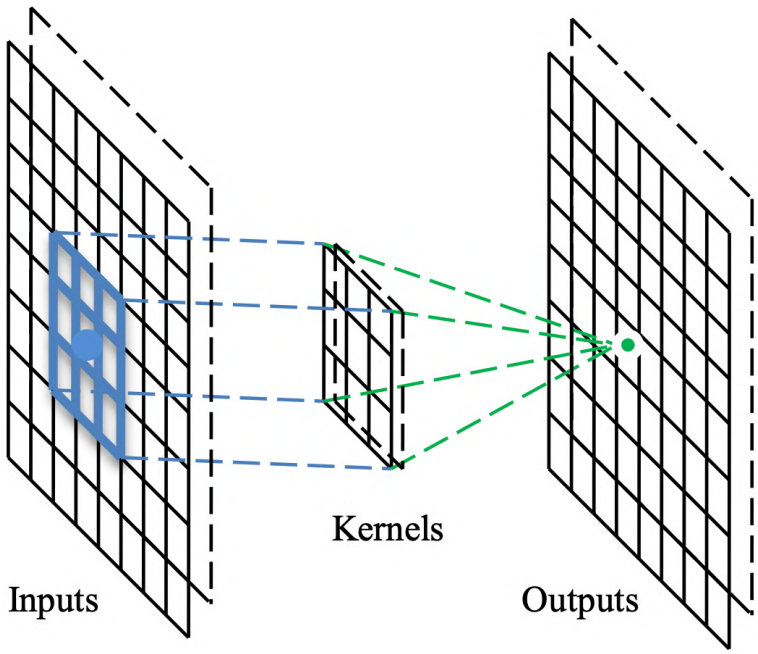


Figure 2.9: The operation of the convolutional layer [43]

A convolution layer takes tiles from an input feature map, applies filtering, and calculates new features, creating an output feature map, as shown in Figure 2.9. The size of the input feature tiles, which typically range from 3x3 to 5x5, allows for the characterization of convolutions by two parameters. The number of filters applied establishes the depth of the output feature map [44].

During the convolution process, filters are positioned to slide across the input feature map in both horizontal and vertical directions, moving one pixel at a time. This method allows the filters to extract and analyze each tile of data, capturing essential features from the input [45]

In a CNN, the network performs an element-wise multiplication between the filter matrix and the corresponding tile matrix for each filter-tile pair. The resulting elements are then summed up to produce a single value representing the output of that particular convolution operation. The convolved feature matrix outputs the resulting values for each filter-tile pair [44].

To incorporate non-linearity into the model, the CNN employs a Rectified Linear Unit (ReLU) layer as an activation function after each convolution operation. The ReLU function, defined as $F(x) = max(0, x)$, returns x for all values of x greater than 0 and returns 0 less than or equal to 0 [42].

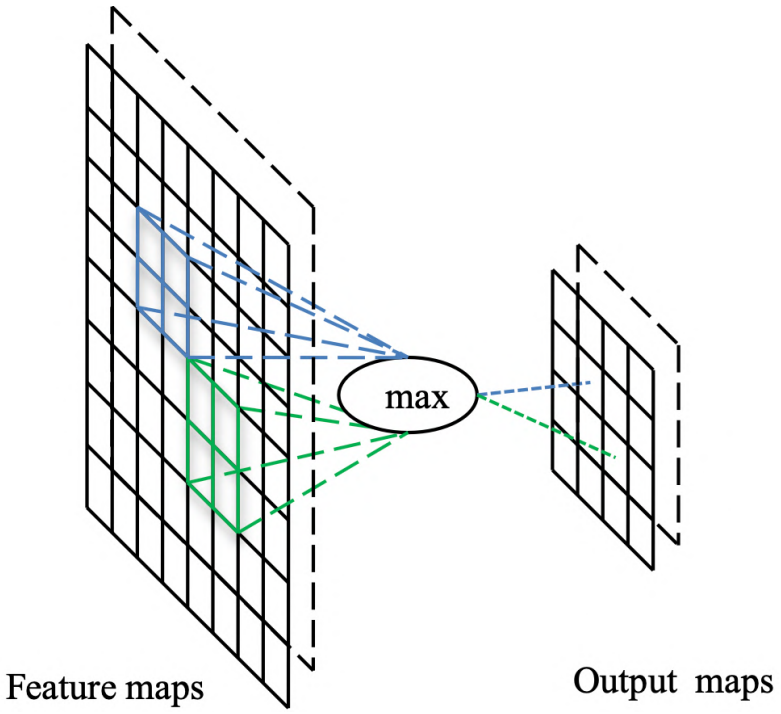


Figure 2.10: The operation of the max pooling layer [43]

The max-pooling layer is applied following the ReLU layer to down-sample the convolved feature. This is achieved by reducing the dimensions of the feature map. This technique is frequently employed to reduce the computational cost and time required [42].

The max-pooling layer works by dividing the feature map into tiles of a specified size, as shown in Figure 2.10. It selects the maximum value for each tile, creating a new feature map with these maximum values while discarding the rest. The max-pooling operation is determined by two parameters: the size of the max-pooling filter and the stride, which dictates how far the filter moves across the feature map [44].

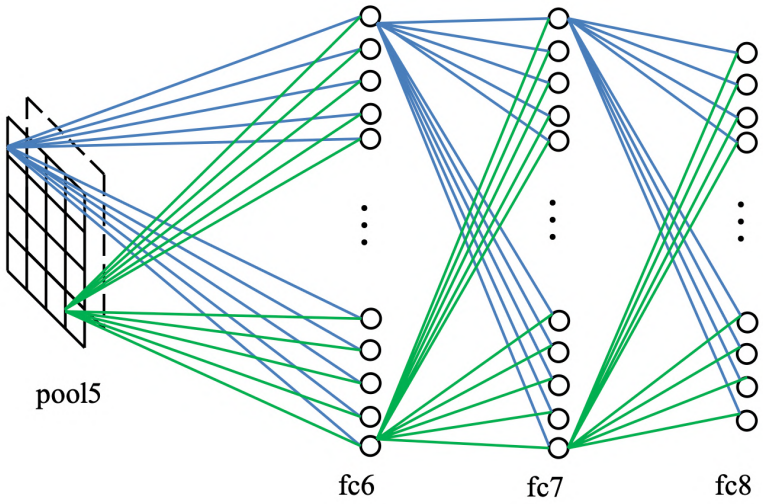


Figure 2.11: The operation of the fully-connected layer [43]

In conclusion, fully connected layers are utilized to execute classification utilizing neurons based on features extracted through convolutional computations. In order to assign probability scores to each label that a classification model predicts, it is necessary to use a softmax activation function in the final layer of the model [42].

This allows a vector of predefined length to be fed forward into the neural network. This vector can then be directed to a specific number of categories for image classification. Alternatively, it can be treated as a feature vector for further processing, as seen in Figure 2.11 [46]. The drawback of these layers is that they involve many parameters, which results in a significant computational cost for training [46].

2.2.6 Overfitting and Underfitting

Overfitting is a common problem in machine learning tasks. It occurs when a learning algorithm becomes too closely attuned to the training dataset, to the point where it starts memorizing the noise or random fluctuations in the data rather than learning the underlying patterns. This causes the algorithm to underperform when tested on an unknown dataset. In this context, the data used in the learning process is critical. Overfitting training data can decrease the model's generalization properties, resulting in unreliable performance when applied to unknown data [42].

To prevent overfitting, data augmentation can be used (see 2.2.7). Conversely, underfitting arises when the model cannot encapsulate the inherent variation within the data, resulting in a classifier with limited capacity to predict outcomes and map training data accurately. This can be attributable to a significant divergence between the training data [47].

2.2.7 Data Augmentation

Larger datasets are commonly thought to improve deep learning models [48]. However, due to the manual effort required for data collection and labeling, creating large datasets can be daunting. For instance, the limited availability of medical image datasets represents a significant challenge in medical image analysis. The rarity of diseases, concerns regarding patient privacy, medical experts' need to label data, and the high cost and manual effort associated with medical imaging processes collectively contribute to the difficulty of building sizeable medical image datasets. [49].

To address this problem, data augmentation increases the variety of available training data without acquiring new data. This is based on the assumption that augmenting the original dataset can extract more information [49].

These augmentations employ data warping or oversampling techniques to expand the size of the training data set. Data warping augmentations preserve existing images' labels, including geometric and color transformations or random erasing. Additional techniques include cropping, padded areas, flipping, color modifications, and the introduction of noise and distortion. The process of oversampling augmentation involves the generation of synthetic instances and their incorporation into the training set [50].

In classic discriminative examples, such as the distinction between cats and dogs, image recognition models must overcome challenges such as viewpoint, lighting, occlusion, background, and scale. Data augmentation aims to incorporate translational invariances into the dataset, thereby enabling models to perform well despite the challenges posed by these invariances [50].

2.2.8 Training, Validation, and Test Data

Partitioning the available data into different subsets for training, testing, and validation is necessary to ensure that models are accurate and generalizable. Training datasets consist of samples used to train models while developing. The model is taught or trained using these collections of examples or patterns. The system uses a training data set to understand the patterns and relationships within the data, thereby learning to make predictions or decisions without being explicitly programmed to perform a specific task [51].

Conversely, validation datasets contain distinct samples to evaluate trained ML models. At this stage, it is still possible to adjust and regulate the model. The model's performance can be assessed, and its parameters can be fine-tuned with validation data. A validation dataset is used to determine how well a model learns and adapts so that adjustments and optimizations can be made to the parameters or hyperparameters of the model before it is tested [51].

While a testing data set is a separate sample, an unseen data set provides an unbiased final assessment of a model's fit. The inputs in the test data are similar to, but not identical to, those in the previous stages. The test data set reflects real-world data the model has never been exposed to. Its main objective is to provide an impartial and conclusive evaluation of the model's performance when it confronts new data in a live, operational setting [51].

2.3 Evaluation Metrics

Evaluating a model's performance is a crucial step in the machine-learning pipeline. It allows us to identify the best hyperparameters for the model and estimate its performance on unseen data in production.

Various evaluation techniques can be employed depending on the method and problem type. Therefore, selecting appropriate evaluation metrics is crucial for distinguishing and obtaining the optimal model. In this section, the focus is on the evaluation metrics of a supervised classification model.

2.3.1 Confusion Matrix

A confusion matrix is a graph that shows the number of actual cases of a given class versus the number of class predictions. It visualizes the results of classifier algorithms and summarizes a model's performance on a given data set [52].

In a matrix, columns represent the predicted values of a given class, and rows represent the class's actual values (i.e., ground truth). Alternatively, rows can be predicted values, and columns can be exact values [53].

The true positives (TP) represent the number of instances in which the model correctly predicts the positive class located in the top left corner. The false positives (FP) in the same column are instances of the hostile class mistakenly identified as positive cases [53].

Similarly, the false negatives (FN) are listed in the top right corner and represent true positive (TP) instances that were wrongly predicted. Lastly, the bottom-right box of the confusion matrix shows the true negatives (TN), which are the instances of the negative class that the model accurately classified as unfavorable. The total number of predictions made by the model can be calculated by summing up all the values in the confusion matrix, including true positives, false positives, true negatives, and false negatives [53][52].

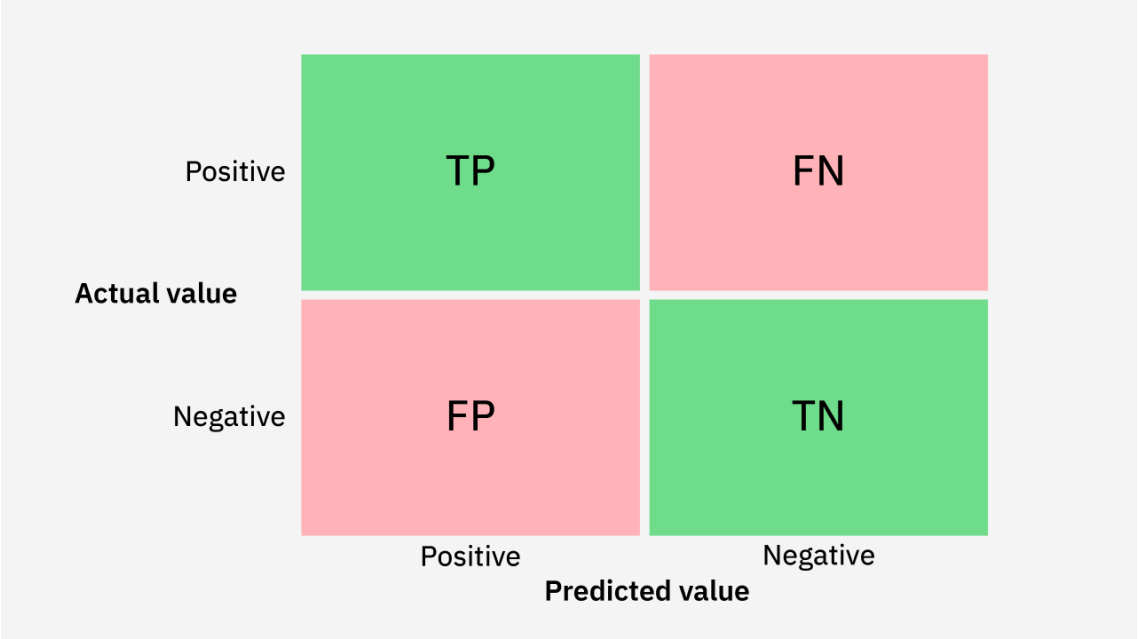


Figure 2.12: Construction of a confusion matrix for a binary classification [53]

Figure 2.12 provides a template for binary classification issues, illustrating how the model's performance can be evaluated in terms of true positives, false positives, true negatives, and false negatives. Similarly, a confusion matrix can be extended to display outcomes for multiclass classification tasks, as depicted in Figure 2.13 [53].

The diagonal boxes illustrate the true positive predictions, while the remaining boxes present the quantities of false positives, false negatives, and true negatives, which vary depending on the selected class [53].

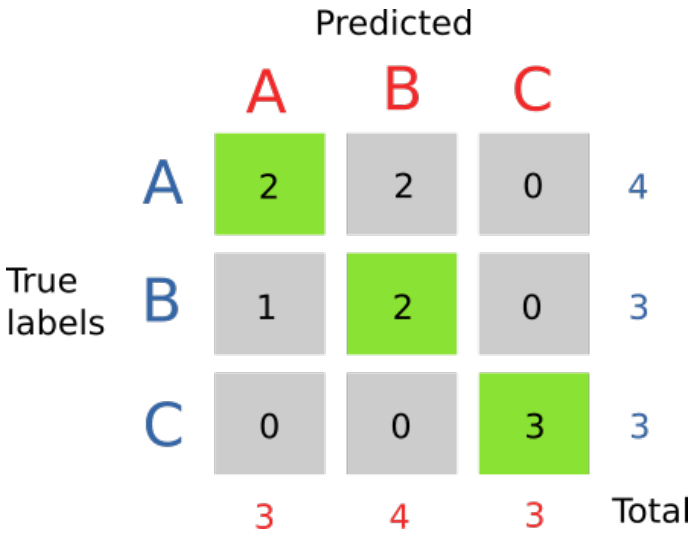


Figure 2.13: Construction of a confusion matrix for a multiclass classification [54]

2.3.2 Accuracy

The accuracy of a trained machine learning model refers to the percentage of correct classifications that it achieves. The metric is calculated by dividing the number of correct predictions by the total across all classes. The resulting value is between 0 and 1. An accuracy score of 0 means that the classifier always predicts the wrong label, while a score of 1 means that it always predicts the correct label.

The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

$$\text{with} \quad TP + TN + FP + FN \neq 0 \quad [55]$$

or put another way,

$$\text{Accuracy} = \frac{1}{N} \sum_{i}^{N} 1(y_1 = y_i') \tag{2.4}$$

where N represents the total number of instances, y_i the actual label of the ith instance and y_i' the predicted label of the ith instance is [56].

Accuracy is a valuable measure for well-balanced classification tasks where all the classes have a comparable number of labels. However, it can hide significant classification errors for classes with few instances that may be less relevant than larger ones. Therefore, it may not be possible to identify the classes where the algorithm is performing poorly [57].

2.3.3 Recall

Recall, a measure of a model's accuracy in correctly identifying instances within a given data set, is also known as sensitivity or true positive rate. It measures the model's ability to capture all the positives and minimize the false negatives [26].

The following formula is used to calculate the recall value:

$$\text{Accuracy} = \frac{\text{Number of true positives predictions}}{\text{Number of true positives and false negative predictions}} = \frac{TP}{TP + FN} \tag{2.5}$$

$$\text{with} \quad TP + FN \neq 0 \quad [58]$$

The higher the recall, the more likely the model effectively identifies the majority of positives and minimizes the chance of missing relevant information. Conversely, a low recall value indicates that the model is not capturing a sufficient number of positive instances, which may result in missed opportunities and potential errors [59].

While high recall is a valuable attribute, it can be achieved at the expense of precision. Models with a high recall value may exhibit a higher rate of false positives, resulting in an increase in false alarms or unnecessary actions. [59].

2.3.4 Precision

Precision is a metric that quantifies the extent to which a model's optimistic predictions align with actual outcomes. It is calculated as the proportion of correctly identified positive instances relative to all the cases that were predicted to be positive [60].

The formula used to compute precision is:

$$\text{Accuracy} = \frac{\text{Number of true positives predictions}}{\text{Number of true positives and false positives predictions}} = \frac{TP}{TP + FP} \tag{2.6}$$

$$\text{with} \quad TP + FP \neq 0 \quad [60]$$

A high precision value suggests that the model is proficient at correctly identifying positive instances while keeping false positives to a minimum. This indicates that when the model predicts a positive class, it is likely to be accurate, thus promoting confidence and reliability in the model's predictions. Balancing precision and recall is essential [61].

Optimizing one metric can negatively impact the other. Strategies to balance precision and recall include threshold tuning and model selection [62]. Precision estimation can be challenging when dealing with imbalanced datasets, as a skewed distribution of classes may lead to inflated precision values [61].

2.3.5 F1-Score

The F1-Score provides a single value that reflects precision and recall, balancing the trade-off between accuracy and recall. It varies between zero and one. A high number signifies a high degree of precision and recall simultaneously, indicating a robust level of performance. Conversely, a low value suggests an imbalance between precision and recall, highlighting areas for improvement in model performance [63]. The value is calculated using the harmonic mean of precision and recall.

The formula represents it:

$$\text{Accuracy} = 2\,\frac{\text{Precision} * \text{Recall}}{(\text{Precision})+\text{Recall}} \tag{2.7}$$

$$\text{with} \quad TP + FP \neq 0 \quad \wedge \quad TP + FN \neq 0 [64]$$

Each class is considered separately to calculate the multiclass, and the score is calculated for each class. Estimating the F1 score for unbalanced data sets can be challenging because the dominant class may bias it. Techniques such as class weighting and resampling, which modify the dataset distribution, can address these issues and ensure accurate performance evaluation [61].

2.3.6 AUROC

The area Under the Receiver Operating Characteristic Curve (AUROC) is calculated by plotting the Receiver Operating Characteristic (ROC) curve. The curve illustrates the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across a range of classification thresholds. TPR measures the proportion of actual positive instances correctly identified by the model [61].

It is calculated as follows:

$$\text{TPR} = \frac{\text{Number of true positives predictions}}{\text{Number of true positives and false negative predictions}} = \frac{TP}{TP + FN} \tag{2.8}$$

$$\text{with} \quad TP + FN \neq 0 \quad [61]$$

A higher TPR indicates that the model effectively identifies positive instances.

The FPR measures the proportion of actual negative instances incorrectly classified as positive by the model. It is calculated as follows:

$$\text{FPR} = \frac{\text{Number of false positives predictions}}{\text{Number of false positives and true negative predictions}} = \frac{FP}{FP + TN} \tag{2.9}$$

$$\text{with} \quad FP + TN \neq 0 \quad [61]$$

A lower FPR indicates that the model effectively avoids false alarms and accurately classifies negative instances [61].

The AUROC is then calculated as the area under the ROC curve and ranges from 0 to 1. Higher values indicate better model performance. An AUROC value close to 1 indicates excellent model performance, meaning the model has high sensitivity and low specificity across all classification thresholds. Conversely, an AUROC value close to 0.5 suggests poor model performance, like random guessing [61]. Like the F1 score, a class imbalance can affect the interpretation of the AUROC, especially if there is a skewed distribution of positive and negative instances [61].

2.4 Introduction to 3D Polygon Mesh and UV-Texture

This subchapter examines the concepts of 3D modeling and texturing, focusing on their application in creating synthetic data for pain recognition. The chapter is divided into three key subtopics: polygonal modeling, texturing, and texture mapping. Each plays a crucial role in developing accurate and realistic synthetic videos.

2.4.1 Polygonal Modeling

3D modeling is a technique that employs the representation of digital objects in three-dimensional space through the use of height, width, and depth. This technique enables the creation of realistic or abstract scenes and objects that can be utilized in many applications, including game development, animation, architecture, and product design [65].

An essential aspect of 3D modeling is positioning objects in space, described by a point or vertex. These points serve as the foundation for the construction of lines and surfaces that define the geometric structure of a 3D model, as shown in Figure 2.14. Edges are the lines that connect two points. While they have a length, they do not yet have a surface [66].

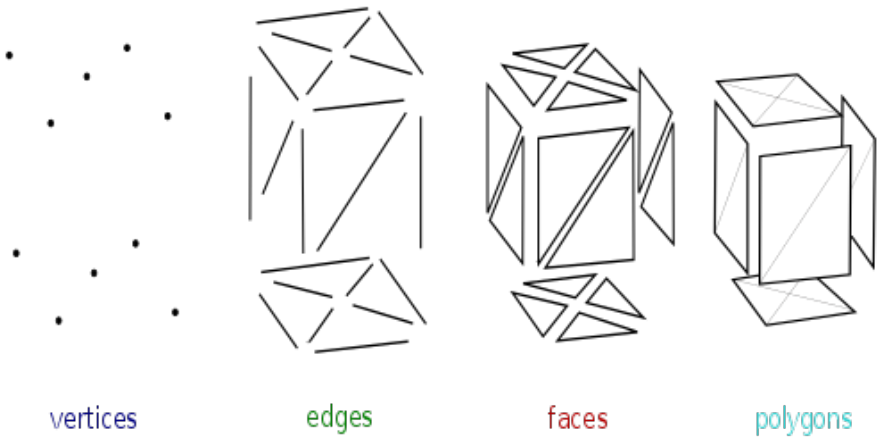


Figure 2.14: Elements of polygonal mesh modeling [67]

Surfaces are created when at least three lines enclose a space and form a polygon. These surfaces are the fundamental elements of the visible surface of a 3D model and give it a three-dimensional shape. Subsequently, several polygons combine to form a three-dimensional object with height, width, and depth information. The number and arrangement of the polygons influence the level of detail and the complexity of the three-dimensional model [65].

One of the most prevalent methods of three-dimensional modeling is polygonal modeling, in which three-dimensional objects are created by moving, rotating, and scaling polygons. Fundamental transformations, such as extruding, mirroring, and subdividing, are standard tools that allow modelers to alter the shape and appearance of an object and adapt it to the desired requirements.

Another necessary process is box modeling, in which simple geometric shapes, such as cubes, spheres, or cylinders, are used as the basis for modeling. These primitives are then modified to create more complex structures [66].

2.4.2 Texturing

Texturing is an essential process in the generation of 3D models. A three-dimensional model is given a surface with specific characteristics in this process. Adding visual details representing the material and surface properties gives the model a realistic or artistic appearance [66].

Textures are bitmap image files described as pixel graphics that provide information about the model's material. They contain color information, patterns, shading, and other visual elements that are applied to the surface of the 3D model. Applying a texture to a three-dimensional object is called texture mapping. The two-stage texturing process begins with UV mapping, where the mesh is told how to receive the visual information. UV coordinates represent positions in the 2D image file, not points in 3D space like XYZ coordinates. The mesh is unfolded along its edges into a two-dimensional representation to project the texture onto the model [65] [66].

Once the texture is created, it is assigned to the 3D model and integrated into a material that unifies all textures. This material determines how the model reacts in terms of lighting and shadows, which significantly impacts the final appearance of the object. Texturing allows for creating realistic and detailed environments and characters for use in games, animations, and other digital media [66].

2.4.3 Texture mapping

Texture mapping is a fundamental aspect of the 3D modeling and texturing process. This process entails the application of two-dimensional textures to three-dimensional models, thereby conferring realistic or artistic surface effects upon them [65].

Textures are two-dimensional images in pixel graphics defined by length and width. They appear to manipulate the surface of the 3D model without affecting the mesh's topology. Textures are measured in texels, representing a pixel in the texture and storing color information [66].

The resolution of the texture, which is determined by the number of pixels in the image file, affects the sharpness and detail of the texture. The higher the resolution, the more detailed and sharp the texture. Realistic details often require the combination of several textures to make an object's height and depth information visible. These structural elements become visible through light perception and shadows, conferring an impressive depth and three-dimensionality upon the model [66].

2.4.4 Open-Source Software Blender

Blender is a 3D software product continuously developed by the Blender Foundation and its users. It allows for creating, texturing, rigging, and animating 3D objects and rendering finished 3D scenes that can be visually modified using compositing. Additionally, Blender includes an integrated game engine that enables the creation of 3D games entirely within the software. It also supports physical

simulations and motion tracking. Blender is open-source software, meaning anyone can use and modify it. Blender is a versatile software comparable to commercial 3D programs like 3ds-Max or Maya [68].

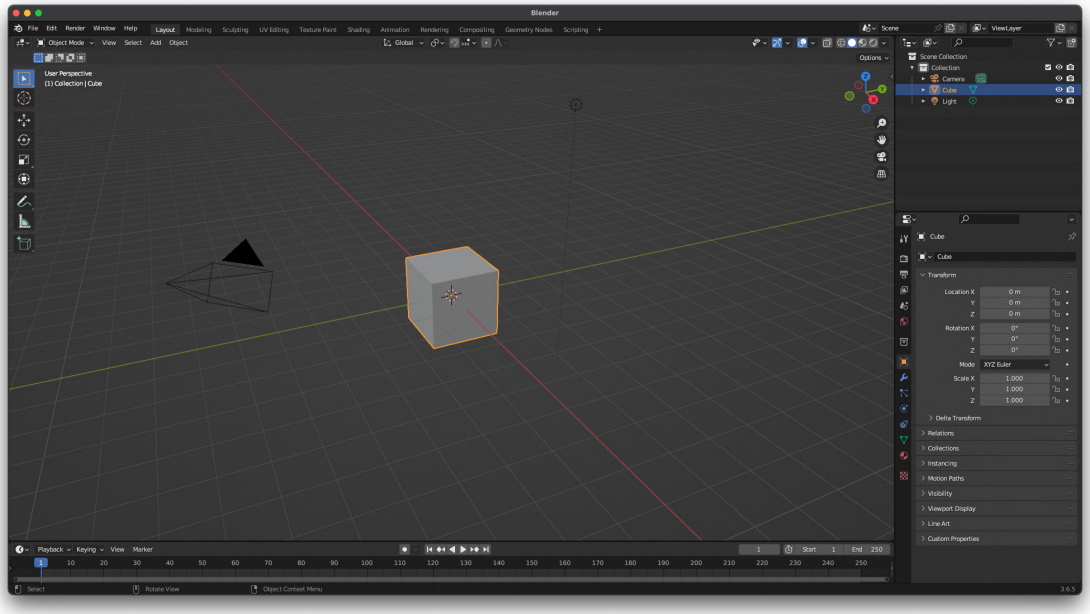


Figure 2.15: Screenshot of the Blender interface

Python scripts or add-ons can be used to modify the software. It is increasingly used in the animation, video game, and research industries. Figure 2.15 displays a cube at the center of the screen. The 3D viewport is the space where objects can be viewed. The tool panel on the left-hand side of the viewport contains tools to modify selected objects. The outliner on the right-hand side of the viewport displays all objects in the scene schematically, and below it is the property panel [68].

3 Related Work

Automatic pain recognition from facial expressions has great potential for healthcare advancements, especially in scenarios where verbal communication of pain is difficult, such as with infants, non-verbal individuals, or those with cognitive decline. This section discusses existing research related to pain recognition, focusing on data collection techniques, learning from synthetic data, pain classification approaches, pain databases, and how our work differs from previous efforts.

3.1 Pain Classification Methods

Early pain classification methods have primarily focused on analyzing features extracted from facial images or video sequences. Martinez et al. [69] employed techniques such as recurrent neural networks (RNNs) to estimate Prkachin and Solomon pain intensity (PSPI) levels from facial images. They also utilized personalized hidden conditional random fields (HCRFs) to assess each individual's visual analog scale (VAS) pain levels.

Haque et al. [70] introduced a novel approach using RGB, depth, and thermal (RGBDT) images for pain-level recognition. They built a database comprising RGB, depth, and thermal images of the face and applied 2D-CNN for frame-feature extraction and pain recognition. Additionally, they employed LSTM networks to find temporal relations between frames and achieve sequence-level pain recognition.

Rodriguez et al. [71] focused on using convolutional neural networks (CNNs) to extract features from the VGG-Faces dataset. This approach was followed by training a long short-term memory (LSTM) network for binary pain estimation (pain or no pain). The study utilized the UNBC–McMaster database of pain detection for its research.

Wang et al. [72] fine-tuned a small pain dataset using a face verification network trained with the WebFace dataset, consisting of 500,000 face images. They approached the pain estimation task as a regression problem, offering a novel perspective on applying machine learning in pain classification.

This work bears similarities to the works of Rodriguez [71] and Haque [70], as they also utilize convolutional neural networks (CNNs) for pain recognition. However, instead of relying on pre-trained networks for emotion recognition, this work employs 3D-CNN for general video recognition tasks. By leveraging the BioVid Heat Pain Database [24], a new data source for pain recognition research is provided. Moreover, this approach uses video data as input, offering a rich source of information for pain-level recognition and analysis.

3.2 Usability of Synthetic Data

Recent advancements have been made in exploring video games as cost-effective and scalable platforms for data collection. Richter et al. [73] [74] utilized Grand Theft Auto V (GTA V) to obtain data and corresponding ground truth for tasks such as object detection, segmentation, and optical flow in driving scenarios. However, they focused primarily on scene and video features rather than facial expressions.

Krähenbühl [75] developed a method using the DirectX 11 API to extract ground truth from video games. This method was applied to games such as Far Cry Primal, The Witcher 3, and GTA V, enabling the extraction of rich image meta-information, including depth and segmentation data.

Roitberg et al. [76] explored the use of life simulation video games, specifically THE SIMS 4, to construct training examples for recognizing Activities of Daily Living (ADL).

This research also uses synthetic data to train a model, similar to previous studies. However, in contrast, it focuses on using synthetic data for facial pain recognition. This approach can potentially advance the field by addressing a unique application of synthetic data in the healthcare domain, explicitly targeting the recognition of patient facial pain.

3.3 Synthetic Data for Pain Recognition

Research using synthetic data to learn pain recognition behaviors is still in its early stages. For example, Pikulkaew et al. [77] utilized the Wasserstein Generative Adversarial Network (WGAN) to generate synthetic images of facial expressions and orientations, but satisfactory results were not obtained as demonstrated in Figure 3.1.

In contrast, this work generates high-quality videos of human meshes that exhibit the same facial gestures while allowing for variable textures. This approach offers a promising new method for learning pain recognition behaviors using synthetic data.

Figure 3.1: Results from GAN of Pikulkaew et al. [77]

3.4 Faces Learned with an Articulated Model and Expressions (FLAME)

Traditional 3D models, constructed from a single scan, accurately represent facial shapes. However, they have limitations in their ability to capture the dynamic nature of facial expressions. (smiling, frowning, etc.). This approach makes subtle differences and variability of real human facial expressions challenging to capture. Furthermore, static models are unsuitable for handling pose variations, such as head tilt or rotation [78].

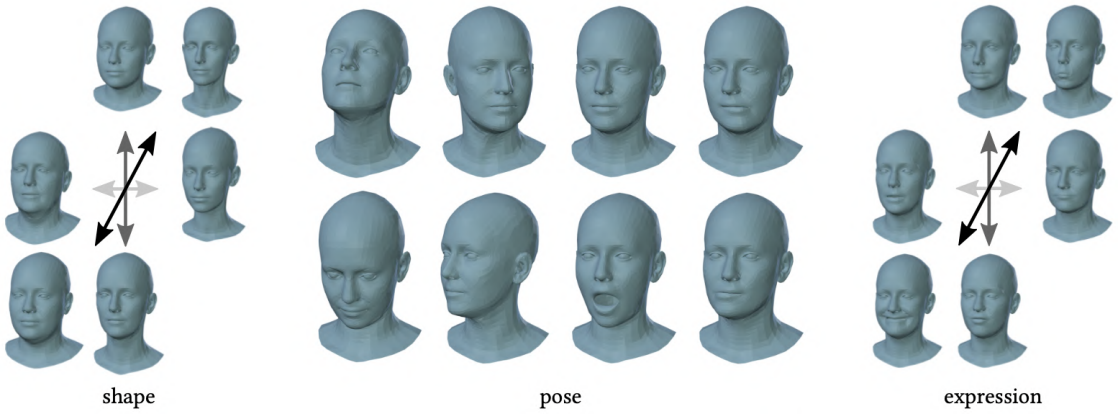


Figure 3.2: Parametrization of the FLAME model [78]

The FLAME model uses 4D scans to capture facial dynamics over time, providing detailed information about facial shape and its changes during expressions. This enables the creation of models representing facial shapes and encoding the underlying mechanisms generating expressions.

FLAME's parametric 3D facial mesh combines critical components such as Linear Shape Space, learned from a large dataset of human heads, and an articulated core with a skeletal structure. This structure includes essential joints of the jaw, neck, and eyeballs, allowing the model to represent pose variations independent of facial expression.

FLAME also utilizes pose-dependent blend shapes to capture subtle changes in facial shape due to pose variations. Additionally, it incorporates global expression blend shapes representing various facial expressions, such as anger, sadness, and surprise, using data from 4D scan datasets like D3DFACS.

The model's parameters, including shape space, pose parameters, and shape weights, are learned from 4D scan data, ensuring that FLAME captures human facial expressions' natural variability and complexity. The model's focus on facial expressions may not capture details like skin texture or wrinkles, and its training data may not encompass the entire range of human facial variations [78].

Furthermore, this work uses the FLAME model for mesh generation, ensuring that every mesh is generated uniformly. This consistency allows for a cohesive representation of various facial expressions and gestures.

3.5 Emotion Driven Monocular Face Capture and Animation (EMOCA)

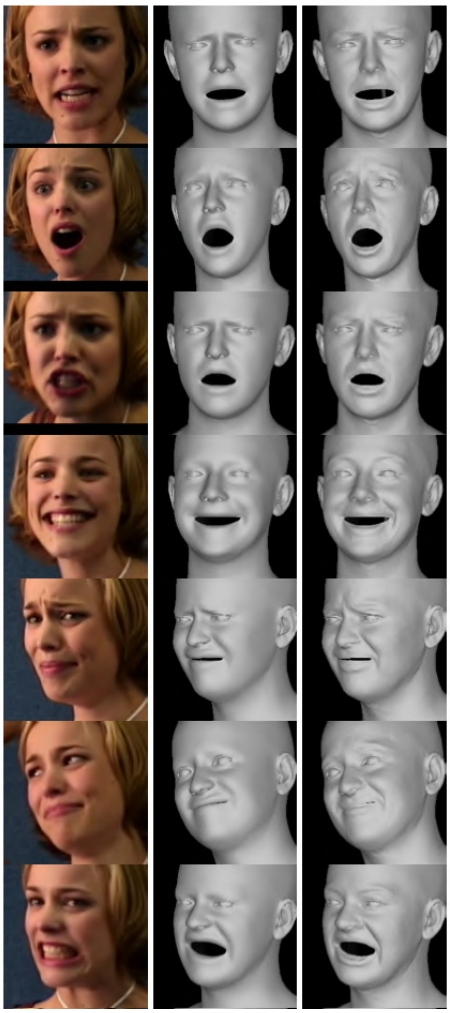


Figure 3.3: EMOCA regresses 3D faces from frames of videos with facial geometry that captures the original emotional content. Left column: images of people with challenging expressions. Middle column: coarse shape reconstruction. Right column: reconstruction with detailed displacements. [1]

EMOCA [1] [2] is a deep-learning framework designed to improve emotion-driven facial animation from a single image and video. It leverages the FLAME mesh for its facial geometry representation, utilizing its detailed linear shape space, articulated core for pose variations, and blend shapes for expressions.

EMOCA's core relies on a deep neural network architecture that processes video inputs and predicts the parameters controlling the FLAME mesh, resulting in a 3D facial reconstruction that reflects the emotion conveyed in the video.

The process includes three stages:

1. **Feature Extraction:**
   Convolutional neural network layers extract high-level features from the input image, capturing information about facial landmarks, textures, and other visual cues crucial for emotion recognition.
2. **Emotion-Aware Regression:**
   A separate network branch classifies the dominant emotion present in the image, such as happiness, sadness, or anger, and predicts emotional intensity, including valence and arousal.
3. **Predicting FLAME Parameters:**
   The network calculates FLAME parameters using the extracted image features and expected emotion information. This includes shape coefficients, pose parameters, and expression blend shape weights, allowing EMOCA to control the FLAME mesh's facial geometry to accurately represent the input image's emotion.

This study uses EMOCA's capacity to input a video and return a mesh for every frame with a comparable head structure, which reduces noise. Rather than using the predefined textures supplied by EMCOA, this study employs textures sourced from FFHQ-UV, which offer a greater range of textures and higher-resolution facial gestures.

3.6 FFHQ-UV: Normalized Facial UV-Texture Dataset for 3D Face Reconstruction

FFHQ-UV [4] is a large-scale dataset of facial UV textures designed to enhance 3D face reconstructions using the FLAME mesh. Traditional methods for facial texture mapping can lead to inconsistencies and artifacts, especially around areas with high geometric detail, such as the eyes and mouth. FFHQ-UV addresses these challenges by providing facial UV textures specifically for the FLAME mesh.

These textures are normalized concerning the FLAME face space, ensuring uniform facial alignment. They capture faces in neutral expressions, minimizing the impact of facial deformations on texture details. Additionally, textures are obtained under uniform lighting to reduce shadows and highlights, improving rendering realism under varying lighting conditions.

FFHQ-UV captures details of skin texture, pores, and variations in pigmentation. Leveraging the FFHQ dataset [79], which contains a vast collection of high-resolution human faces, FFHQ-UV uses StyleGAN-based facial image editing to generate multi-view, normalized images from single-image inputs.

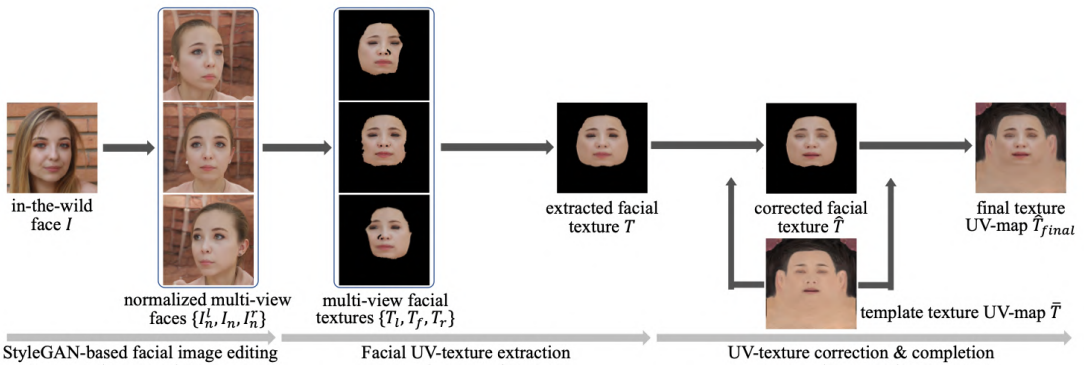


Figure 3.4: Pipeline for producing normalized texture UV-map from a single in-the-wild face image [4]

The creation of FFHQ-UV involves a complex processing pipeline, as displayed in Figure 3.4. Firstly, StyleGAN-based Image Editing is used to manipulate facial images in FFHQ. This manipulation includes generating multiple views of a single face while maintaining a neutral expression and consistent lighting conditions, using StyleGANs, a generative adversarial network (GAN).

After that, UV-texture extraction and correction are performed. Appropriate UV maps are extracted from the generated multi-view images to ensure compatibility with the FLAME mesh. Any inconsistencies or irregularities in the extracted textures are automatically corrected.

In some cases, certain areas of the facial texture might be missing due to limitations in the image editing process. FFHQ-UV employs techniques to complete these missing regions while maintaining texture consistency to address this issue.

Finally, the extracted and potentially completed textures undergo three processing stages, where correction and reduction occur to prepare them for use with the FLAME mesh, as seen in Figure 3.5.

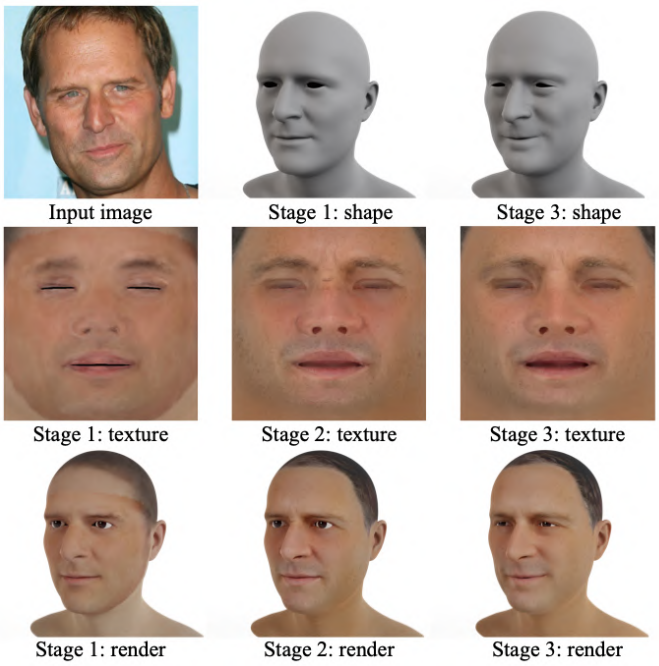


Figure 3.5: Intermediate reconstruction results of each of the three stages [4]

This work uses the FFHQ-UV generation step to produce textures but also modifies face textures to include eye textures for the eye mesh, as the standard textures don't generate them. This modification ensures a more comprehensive and realistic representation of the entire face.

3.7 Objectives

This work differs from previous pain recognition research in its innovative approach to data collection. While prior efforts have focused on real human video data or limited synthetic data generation, this study uses a large dataset of synthetic facial expressions based on the BioVid Heat Pain Database. Using synthetic data allows for the practical training of a robust pain recognition model that can mitigate the challenges and limitations associated with real-world data collection.

Furthermore, the trained model is integrated with a real-time pain classification system that analyzes facial expressions in real human videos. This integration helps to bridge the gap between synthetic data training and real-world application, which could lead to a more practical and scalable approach to pain recognition.

In conclusion, this review has explored existing research on pain recognition. The review highlights data collection methods, learning from synthetic data, pain recognition approaches, and pain databases.

The novelty of the work is emphasized by showcasing its unique utilization of synthetic data for training and its subsequent integration with a real-time pain recognition system. This approach can potentially advance automated pain recognition and its applications in healthcare settings.

4 Dataset Generation

This chapter outlines the pipeline for generating synthetic head videos based on the BioVid Heat Pain Database [24], as shown in Figure 4.1. First, input videos are extracted into individual frames. For each frame, a head mesh is generated using EMOCA [4]. Concurrently, textures are created from an input image using FFHQ-UV. Finally, the frames of the head meshes with textures are rendered into a synthetic head video using Blender.

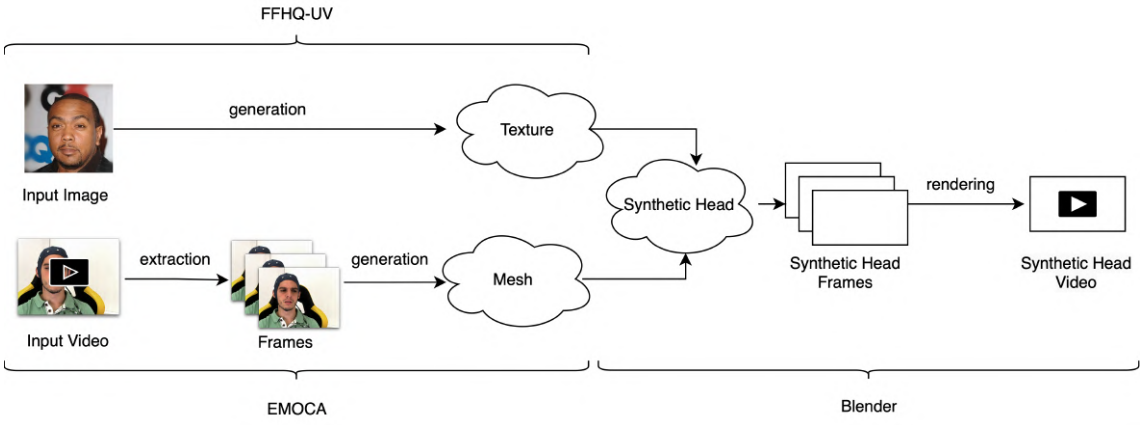


Figure 4.1: Generation Pipeline of Synthetic Head Videos

4.1 Experimental Data

The BioVid Heat Pain Database [24] is used to create a set of synthetic data. The BioVid Heat Pain Database contains data from 90 participants who underwent thermal stimulation to induce pain. Figure 4.2 shows some facial samples from this database.

Before recording data, individual pain thresholds and tolerances were determined to establish a baseline for pain intensity levels. The main experiment included four pain intensities (PA1 to PA4), chosen based on individual thresholds and tolerances, with intermediate intensities (PA2 and PA3) being linearly interpolated between PA1 and PA4.

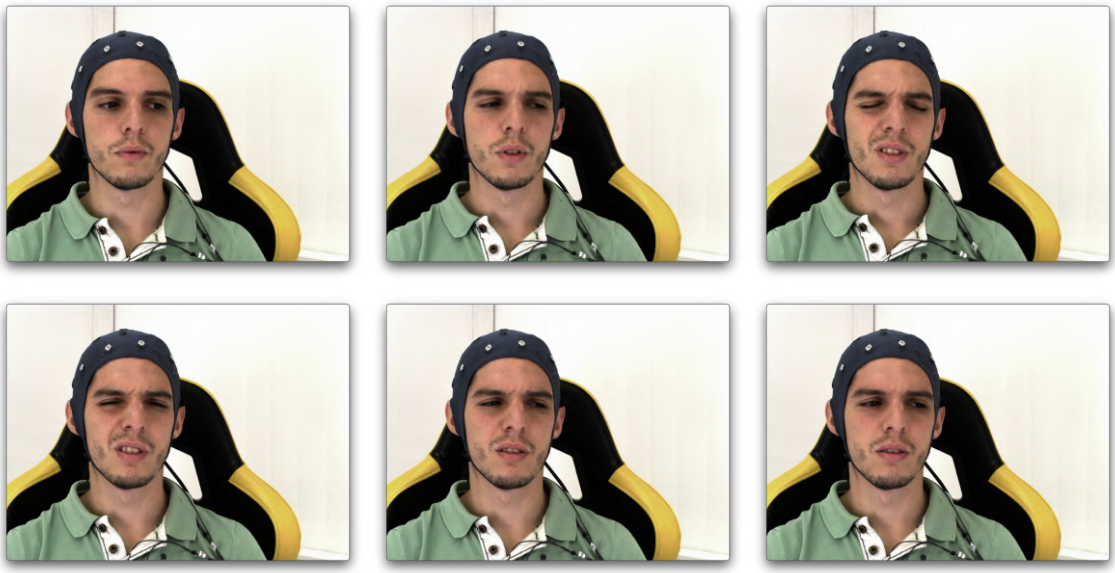


Figure 4.2: Face samples from the BioVid Heat Pain Database

Each intensity level was stimulated 20 times, with pauses used to obtain non-painful baseline samples (BLN). One of the primary analyses performed on the BioVid Database was evaluating facial activity in response to pain stimulation. Median facial activity was compared among pain stimulation classes, revealing distinct patterns.

Increased facial activity was observed in PA4 and PA3, indicating an exaggerated pain response, shown in Figure 4.3. Conversely, activity in PA2 was only slightly above baseline levels, suggesting a milder pain experience. Rare facial reactions were observed in low-intensity pain stimuli (PA1 and PA2), whereas higher scores were more prevalent in response to elevated pain intensities (PA3 and PA4).

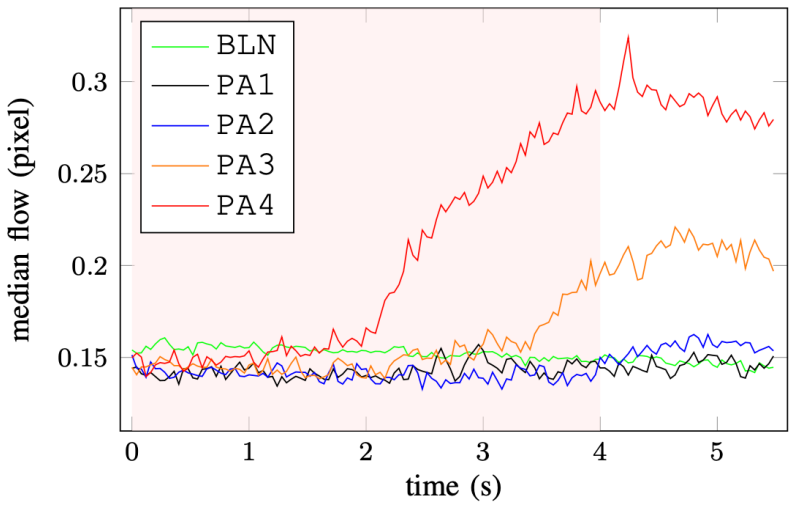


Figure 4.3: Median flow time series during pain stimulation (PAn) pauses (BLN). The red background illustrates the timing of the high-temperature plateau for stimulation [80]

This distribution highlights the association between pain intensity and perceived pain sensitivity. According to Figure 4.4, facial reactions appear infrequent in low pain intensities. It is worth noting that only a tiny percentage of participants in PA1 and PA2 had a maximum PSPI score greater than zero.

However, it is essential to consider that PSPI=1 may not accurately reflect the participants' pain levels, as many had their eyes closed during the study, which could have affected the results. Based on these findings, it can be inferred that higher pain intensities are linked to higher scores [80].

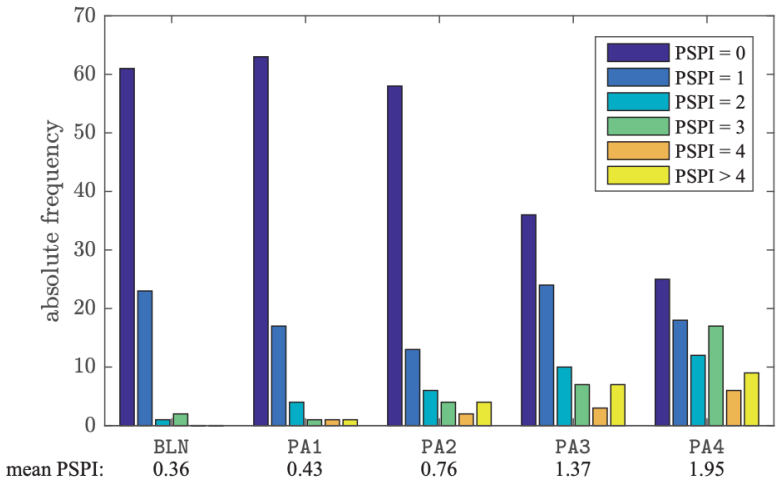


Figure 4.4: Distributions of the maximum PSPI across pain intensities [80]

For many subjects, the stimulation did not exceed the critical thresholds to trigger a facial response in most of the PA1 and PA2 samples. Therefore, the focus was shifted to the baseline BL1 videos and the PA3 / PA4 videos only to avoid any noise that might interfere with the machine learning algorithms used for automatic pain detection, as was done by Werner et al. [81].

4.2 Mesh Generation

The EMOCA framework simplifies mesh generation by using the input video data. The video data is segmented into individual frames, and each frame is processed to isolate the head region. This process lays the foundation for generating the head mesh and allows for extracting key facial features. Using this approach, a consistent mesh structure can be generated across multiple frames. The mesh generation process starts with the input image being fed into various encoders.

The encoders process the input sequence of images to generate a set of context vectors. These vectors encapsulate essential information about factors such as expression, shape, pose, and other parameters relevant to constructing the 3D mesh. Once the context vectors are obtained, the FLAME geometry provides a flexible and parametric representation of 3D facial geometry, making it an ideal choice for mesh reconstruction.

The encoded representation produced by the encoders is translated into a detailed 3D mesh structure with the help of a decoder. The decoder is a crucial component in the mesh generation pipeline, responsible for reconstructing the encoded representation into a tangible 3D mesh. By interpreting the context vectors generated by the encoders, the decoder effectively translates these abstract representations into concrete geometric structures, forming the basis of the final mesh.

The EMOCA framework offers a key advantage in mesh generation by providing consistency across generated meshes. This is achieved by utilizing a standardized head structure derived from the input data, resulting in high uniformity per video. This uniformity minimizes noise and inconsistencies, ultimately enhancing the quality of the rendered synthetic video.

4.3 Texturing

The FFHQ-UV repository [4] provides a reliable framework for generating textures for synthetic heads. The process starts with facial landmark detection, which is a crucial step. Convolutional neural networks are utilized to thoroughly analyze the input image, identifying vital facial features such as eyebrows, eyes, nose, and mouth. The recognized landmarks are crucial reference points for the following stages in the workflow, as illustrated in Figure 4.5.

Figure 4.5: Texture extraction steps

Once the landmarks are detected, a synthetic face is created using them. This forms the initial structure for the synthetic head. The FFHQ-UV approach then performs texture mapping. This process accurately applies high-resolution textures onto the underlying synthetic head mesh. This mapping process is essential in giving the synthetic face a lifelike appearance, turning it from a generic base into a visually appealing representation of a human head.

It is worth noting that the automatic mapping script provided by the FFHQ-UV repository, while efficient in mapping textures onto the synthetic head mesh, may overlook a crucial detail of the eye textures. This refinement addresses a limitation of synthetic heads with blank eye textures, which can detract from the overall level of realism.

Eye-specific textures were carefully added to the workflow to ensure optimal visual fidelity. These eyes will match the eyes on the synthetic head form. This manual intervention ensures that the synthetic head's final appearance has realistic and diverse eye textures, enhancing the visual fidelity of the dataset.

The existing script was also modified to promote consistency and optimize workflow efficiency. This modification allows the script to handle this step for all frames within a dataset automatically. This automation can be beneficial when dealing with large-scale datasets, such as the BioVid Heat

Pain Database's 8600 video headshots, as it reduces manual effort and streamlines the texture creation process. The selected textures are derived from actual individuals featured on CelebHQ [3], exhibiting a significant level of variety regarding age, gender, and ethnicity.

4.4 Rendering

The Stop Motion OBJ plugin [82] was used for the Blender render. This plugin acts as a connection between Blender and mesh sequences, allowing the importation of a sequence of mesh files and rendering them as an animation without interruption. Using Stop Motion OBJ, the mesh sequences were imported into Blender, establishing the foundation for the rendering process.

The imported mesh sequence is automatically processed using Blender scripts. The scripts facilitate tasks such as texture application and animation setup, streamlining the rendering workflow. Automating these processes allows for consistent and efficient rendering of synthetic videos, saving time and effort in the production pipeline.

Rendering in Blender allows experimentation with different camera perspectives. The synthetic videos adopt two main perspectives: a frontal view and a side view of the head. These perspectives present varying viewpoints, contributing to the visual complexity and depth of the rendered animations. The resulting synthetic videos are significant because of the high-resolution rendering and intricate geometry. Rendering a single perspective of the video results in a file size of approximately 18 GB.

A cluster system was utilized to optimize the rendering process and manage the computational workload effectively. The rendering workload was distributed across many CPU cores using multiple nodes and threading techniques. This resulted in running 20 nodes with 16 threads per node for 320 concurrent threads. Optimization efforts have significantly improved rendering efficiency. Fine-tuning and optimizing rendering parameters reduced the rendering time for 8600 videos to approximately 2 hours.

This optimization leads to an average render time of roughly 5 minutes per video, facilitating the rapid production of synthetic video, as seen in Table 4.1. It is important to note that the render time per video depends on the CPU used. For instance, with an Apple M1 chip, the render time was only 1 minute per video, which reduced the total sequential processing time to approximately six days.

Strategy	Time Adjustment	Total Time
Sequential Processing	8600 Videos $*$ ~5 min per Video	~654 hours / ~27 days
Using multiple nodes (20 nodes)	654 hours / 20 nodes	~33 hours
Using multiple threads (16 threads)	33 hours / 16 threads per node	~**2 hours**

Table 4.1: Optimization steps for Synthetic Head Video Generation

4.5 Challenges

1. **Generating Meshes and Textures:**
   One of the initial attempts to generate meshes for the BioVid Heat Pain Database was to use the DAD-3DHeads [83] repository. However, this method lacked the capability for automatic UV mapping, a crucial step in applying textures to the mesh.

   As a result, texture application for the entire database became a complex and potentially impossible task. Furthermore, DAD-3DHeads exhibited inconsistencies in mesh alignment and orientation, which increased the error rate for potential UV mapping attempts.

   To overcome the limitations of DAD-3DHeads, a new workflow was implemented. This involved utilizing StarGAN-V2 [84] to generate a fusion of two distinct human faces for each video frame. Subsequently, FFHQ-UV was used to create a mesh with the corresponding texture.

   Although this approach provided per-frame texture generation, it resulted in a significant time bottleneck, generating a 3D mesh with a StarGAN texture taking approximately 1 minute per frame. The processing time required for a video with 137 frames at 25 FPS is 2.17 hours. Scaling this to a dataset of 8600 videos would take 778 days or 26 months to process without considering potential threading or parallelization techniques.

   Additionally, the generated meshes for each video frame were inconsistent, resulting in facial proportions and size variations. The inconsistency introduced noise into the synthetic video sequence.

   Using EMOCA to generate the mesh reduced the time needed to create the mesh sequence to approximately 3 minutes. EMOCA also ensures consistent facial proportions and sizes, reducing noise in the synthetic video sequence. Additionally, since only a limited pool of textures was needed, generating new textures for each frame was unnecessary, saving significant time.

2. **Rendering:**
   PyTorch3D [85], a popular Python package, was initially used for rendering meshes with textures. However, PyTorch3D is not explicitly optimized for the high accuracy required to render human faces with intricate details like facial expressions. This resulted in limitations in accurately capturing the subtleties of facial gestures.

   In contrast, Blender's render engine allows users to create high-resolution 3D animations quickly and efficiently, offering a more time-effective alternative to PyTorch3D.

The current workflow utilizes EMOCA for mesh generation, FFHQ-UV for UV mapping, and Blender for rendering. This method offers a valuable trade-off between rendering speed and accuracy, significantly reducing noise in synthetic videos compared to the StarGAN-based approach. It also facilitates faster and more efficient rendering, enabling the capturing of intricate details in facial expressions within videos.

4.6 Limitations

Despite advancements, certain limitations remain inherent to the current mesh generation techniques. Minor discrepancies can occur between facial landmarks identified in the image and their corresponding positions on the projected 3D model. This can lead to inconsistencies in the final rendered video.

Moreover, the generated models lack hair, leading to a static appearance across various videos. Ideally, the generation process should dynamically create hairstyles to enhance realism.

Additionally, eye movements are not yet incorporated, with eyes remaining fixed and front-facing in all rendered videos. This limitation restricts the expressiveness of the synthetic faces and hinders their ability to convey emotions effectively.

Furthermore, fine details like freckles or faint wrinkles might be smoothed out during the texture generation process in the FFHQ-UV repository. This can result in a loss of visual richness in the synthetic faces, detracting from their overall realism.

4.7 Dataset Creation

The generated datasets are sectioned into two distinct categories, each sorted according to different data quality and quantity aspects. The first dataset category focuses on the influence of transitioning from natural to synthetic data. It aims to clarify whether synthetic data can reproduce the performance achieved with accurate data.

The second category explores the impact of different data situations, explicitly focusing on the quantity of data available for training. Through systematic partitioning of these two dataset dimensions, valuable insights into synthetic data's potential advantages and drawbacks in pain recognition based on synthetic data will be provided.

4.7.1 Effect of Video Quality

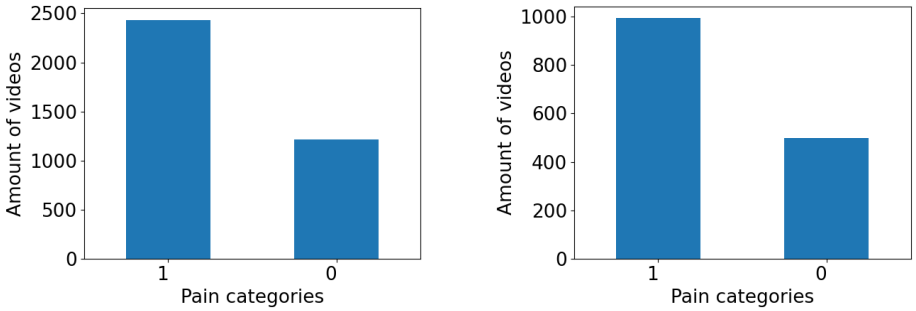
To describe the impact of video quality on pain detection, the following datasets are generated, ranging from real videos to synthetic data with various textures and viewpoints.

1. **Meshes without Texture in Frontal View:**
   Dataset where meshes are generated without texture.
2. **Meshes with Texture in Frontal View:**
   Texture is applied to meshes in a frontal view using textures from a single male individual to enhance consistency.
3. **Meshes with Texture in Side View:**
   Dataset with Synthetic Heads from a lateral recording. As in dataset two, a texture from a single individual (male) was utilized for consistency.

4. **Random Assignment of Synthetic Heads to Videos:**
   All videos in the dataset are randomly assigned synthetic heads, replacing the original individuals in the video.

5. **Meshes with Textures in Front and Side View:**
   Dataset containing combination of front and side-view textures.

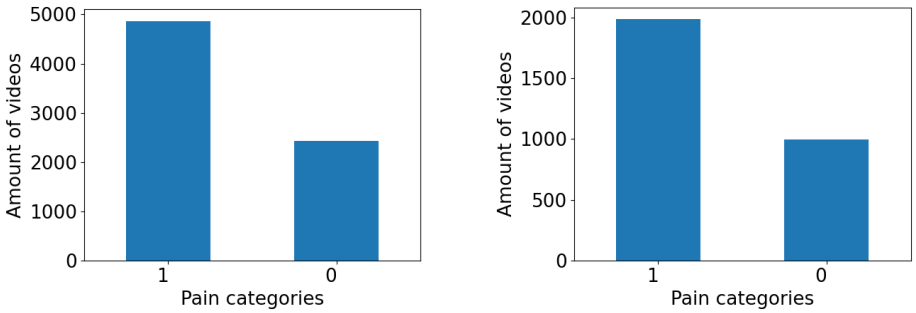
The number of videos in the datasets required to train and validate the model may vary depending on the use case. As depicted in figure 4.6, datasets 1-4 have the same number of videos in the data sets, as only the type of videos used in each scenario differs (real videos, synthetic videos, and different perspectives).



(a) Distribution of training data **(b)** Distribution of validation data

Figure 4.6: Training and validation data distribution for datasets 1 - 5. Label 1 indicates the presence of pain, while label 0 indicates its absence.

Conversely, dataset 5 requires twice as many videos due to the different perspectives, as seen in 4.7. Regardless of the total number of videos, the distribution of videos per dataset is 2:1 for both the training and validation data.



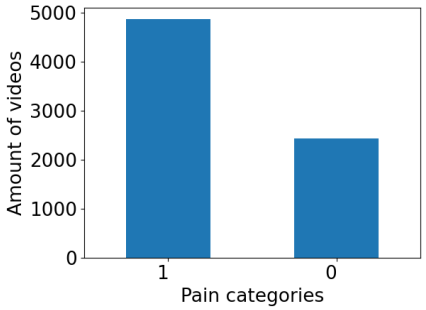
(a) Distribution of training data **(b)** Distribution of validation data

Figure 4.7: Training and validation data distribution for dataset 6. Label 1 indicates the presence of pain, while label 0 indicates its absence.

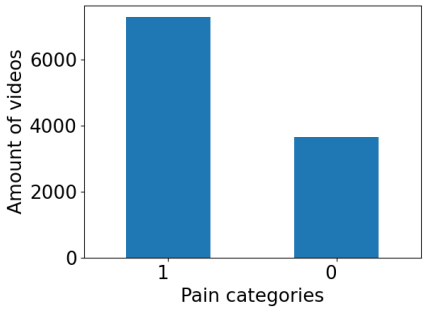
4.7.2 Effect of Video Quantity

The following datasets are generated to examine the influence of data volume on pain detection, focusing on assessing whether increasing the volume of synthetic data can improve model performance relative to real data.

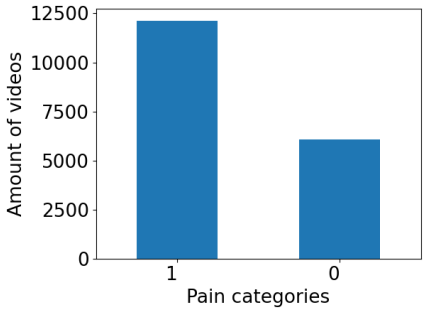
1. **Replacement each Test-Patient with One Synthetic Head Texture:**
   This dataset replaces each BioVid Heat Pain Database patient with a single synthetic head texture. Due to simply altering the texture of the synthetic head, the dataset distribution remains the same as shown in Figure 4.6.
2. **Replacement each Test-Patient with Multiple Synthetic Head Textures (2, 3, 5, 10):**
   The dataset contains multiple synthetic head textures for each individual, ranging from 2 to 10 synthetic heads per individual.



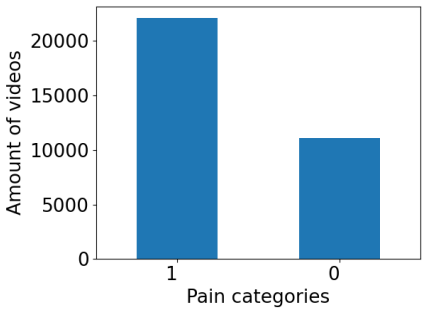
(a) Replacement of **2** different Head Textures per Test-Patient



(b) Replacement of **3** different Head Textures per Test-Patient



(c) Replacement of **5** different Head Textures per Test-Patient



(d) Replacement of **10** different Head Textures per Test-Patient

Figure 4.8: Distribution of training data's replacement of each Test-Patient with Multiple Synthetic Head Textures (2, 3, 5, 10). Label 1 indicates the presence of pain, while label 0 indicates its absence.

As shown in Figure 4.8, the number of videos in the datasets is increasing linearly. The dataset with two replacements (Figure 4.8a) contains approximately 2500 videos, while the dataset with ten replacements contains approximately 20000 videos (Figure 4.8d).

This selection process prioritized diversity in ethnicity, age, and gender. By carefully curating the textures, the goal was to create a dataset that reflects the broad spectrum of human appearance, fostering a more generalizable and inclusive pain recognition model, as illustrated in Figure 4.9.

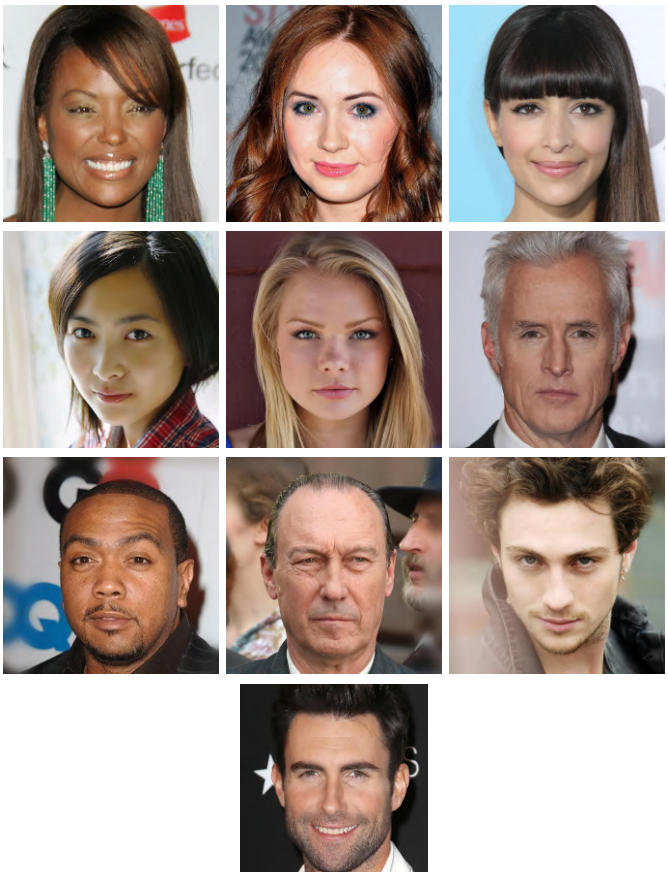
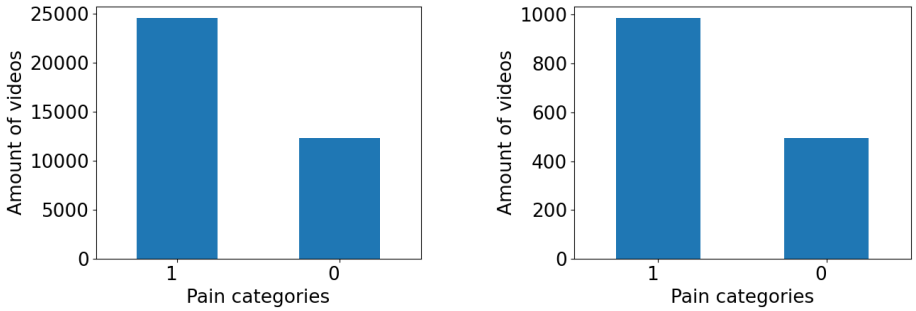


Figure 4.9: CelebHQ face samples used for the generation of the texture

3. **Synthetic Data + Real Data:**
   This dataset combines synthetic and real data from the BioVid Head Database.



(a) Distribution of training data **(b)** Distribution of validation data

Figure 4.10: Distribution of training and validation dataset for scenario . Label 1 indicates the presence of pain, while label 0 indicates its absence.

In addition to the dataset comprising all ten synthetic human heads, the BioVid Heat Pain videos were also taken into account, resulting in nearly 25,000 videos for training purposes, as seen in Figure 4.10a. All validation videos were randomly mixed to avoid overstressing the validation step, with only 1,400 chosen for analysis (Figure 4.10b).

5 Models for Pain Recognition

This chapter introduces the model architecture used to evaluate the feasibility of synthetic video, as shown in Figure 5.1. Initially, videos undergo augmentation to focus on the most critical areas and normalize the data. The training data is also wrapped to increase the training set's size, making it more diverse and representative. After every 10 epochs of training the SlowFast-R50 model, the model's progress is assessed using the validation set. This allows for a temporary evaluation of the model's performance and saves a checkpoint of the model's current state, helping to monitor and preserve successful model configurations.

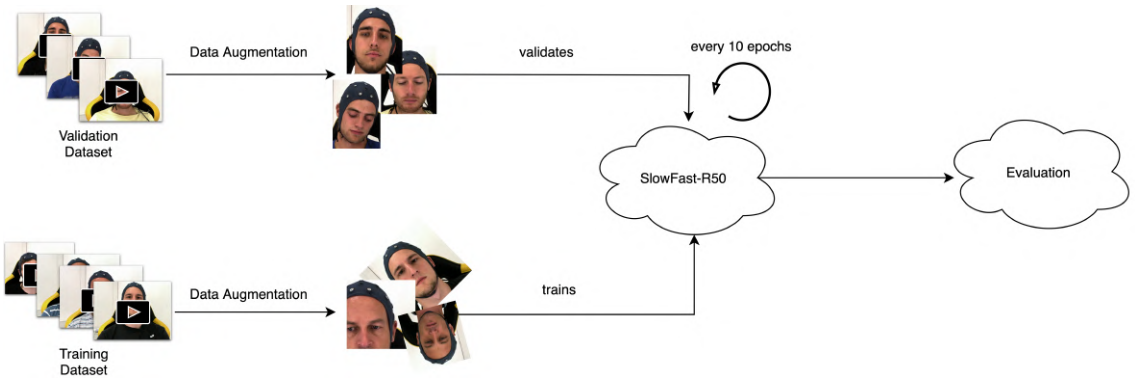


Figure 5.1: ML Modeling Cycle

5.1 Deep Learning Model Design

Previously, pre-trained networks for emotion recognition have commonly been utilized for pain recognition tasks. However, it has been observed that these networks may not always provide adequate supervision during optimization, which can result in artifacts in the reconstructed geometry [86] [1]. A network designed explicitly for general video recognition tasks was chosen to address this limitation.

Among the available video recognition networks, such as I3D [87], Slow-R50 [88], and C2D [89], the SlowFast-R50 network [88], developed by Facebook Research, appears to be the most promising option. It has demonstrated superior performance across various metrics, particularly regarding the AUROC score.

The ResNet-50 architecture is at the core of the SlowFast-R50 network. ResNet, short for Residual Network, greatly influenced convolutional neural network (CNN) architectures by introducing the concept of residual learning. This approach utilizes skip connections to address the vanishing gradient problem, a common challenge encountered during the training of deep neural networks, especially in architectures with many layers [90] [91].

During training, it has been observed that the gradients used to update the network's parameters may diminish significantly as they propagate backward through the layers. As a result, there may be slow or no updates to the weights of early layers, leading to slow convergence or stagnation in learning. This phenomenon can potentially hinder the network's ability to learn complex representations effectively, which may limit its performance [91].

ResNet50 is a specific model within the ResNet architecture family, known for its 50-layer deep network design. It includes various layers, such as convolutional, pooling, and fully connected layers. ResNet50 consists of 48 convolutional layers and includes other layers, such as pooling and fully connected layers, to process and transform the input data [90].

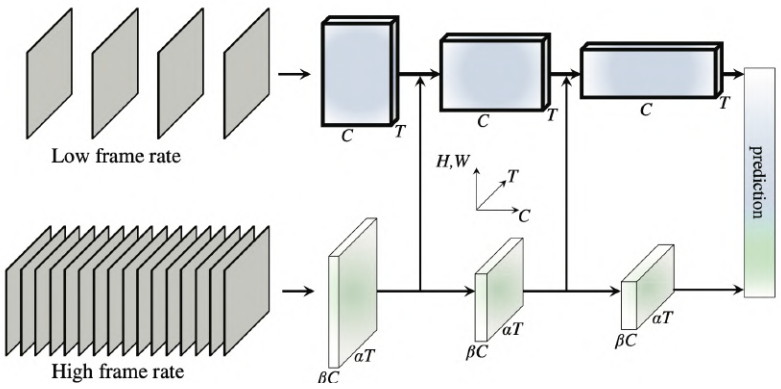


Figure 5.2: A SlowFast network with a low frame and high frame rate with information exchange after each stage [88]

The SlowFast-R50 network is distinguished by incorporating two pathways, as shown in Figure 5.2. These pathways operate at different temporal resolutions and are designed to capture distinct aspects of video data. The slow path is intended for processing static or slowly changing scenes, focusing on capturing semantic information over longer time scales. Furthermore, it operates at a lower frame rate and slower refreshing speed [88].

In comparison, the fast pathway operates at a higher frame rate and quicker refreshing speed, enabling it to capture fine-grained motion details. Throughout the network architecture, the information of the slow and fast pathways is fused to ensure that each path remains aware of the representations learned by the other. This fusion process occurs after every stage in the network, facilitating the integration of semantic and motion information at multiple levels of abstraction [88].

The SlowFast-R50 network is a powerful solution for pain recognition tasks, achieved by leveraging the ResNet-50 backbone and incorporating dual pathways for processing video data. It captures semantic and motion information at different temporal resolutions, making it well-suited for analyzing dynamic and complex video data, such as facial expressions indicative of pain [88].

5.2 Implementation and Training Details

The models were trained for a total of 100 epochs in all scenarios. Network architecture parameters, such as dropout and input size, were configured according to the specifications of the original architecture. Notably, the last layer was modified to have only one output node, facilitating binary pain recognition.

Diverse data augmentation techniques were utilized during training to enhance model generalization and robustness. Subsampling involved uniformly sampling frames from the input video, while video normalization ensured consistent pixel values across RGB channels. Random short side scale and random cropping introduced video size and content variability, while random horizontal flip augmented dataset diversity. Additionally, a pathway function transformed video frames into two lists of tensors, effectively capturing temporal and spatial information.

Similar augmentation techniques, including subsampling, normalization, short-side scaling, center cropping, and pathway function application, were applied during validation to prepare the validation data for model evaluation.

The Stochastic Gradient Descent (SGD) optimizer was employed for training, with a learning rate of 0.01, weight decay of 1e-5, and momentum of 0.9. These parameters were chosen to control the step size during gradient descent, prevent overfitting, and accelerate convergence by incorporating past gradients.

To ensure representative training and validation datasets, data splitting followed a strategy proposed in previous research [92]. This strategy aimed to maintain similar gender, age, and expressiveness distributions between the training and validation sets.

A batch size of 64 was selected for training, balancing computational efficiency and model stability. Due to an imbalance in the data, there are more videos with the label ”pain” than with the label ”no pain” (see chapter 4.7). To address this, a Weighted Random Sampler was employed, which ensures that the number of videos with each label is distributed evenly across epochs. The loss function (BCEWithLogitsLoss) was also utilized, and a weight of 4 was added to the positive examples. This weight serves to compensate for the positive-to-negative sample imbalance.

6 Experimental Setup

The research focuses on two distinct categories that assess data quality and quantity to investigate the effectiveness of synthetic data in recognizing pain using the model described in Chapter 5 and the datasets in Chapter 4.7. This approach aims to provide valuable insights into synthetic data's potential advantages and drawbacks in pain recognition.

The experimental procedure is as follows:

1. **The establishment of research cases for Data Quality and Quantity:**
   Partitioning research cases into two categories based on data quality and quantity. Analyzing these two aspects will help comprehend how synthetic data compares to real human data regarding pain recognition performance.
2. **Training and Validation:**
   The models are trained and validated within their respective domains using the synthetic data. This process helps fine-tune the models to achieve optimal performance with the synthetic data provided.
3. **Testing with Real Human Dataset:**
   Once trained and validated, the models are tested using the real human dataset from the BioVid Heat Database. This step evaluates how well the models, trained on synthetic data, perform when applied to real human data.

By systematically analyzing these aspects, the research aims to understand how synthetic data can enhance or hinder pain recognition compared to real human datasets. This comparison will provide valuable insights into the effectiveness of synthetic data in pain recognition and its potential applications in the field.

6.1 Research cases

The following describes the different research cases, distinct into data quality and quantity categories. It is also important to note that when the perspective is not explicitly stated, it is assumed to be the frontal of the synthetic heads.

6.1.1 Effect of Video Quality

- **Base Case BioVid Heat:**
  The BioVid Heat Pain Database is utilized as the benchmark for pain recognition, establishing a standard against which the performance of synthetic data is measured. This approach aims to determine the accuracy and reliability of pain recognition models in real-world settings.

- **Only Mesh:**
  This research case uses the dataset ”Meshes without Texture in Frontal View” to investigate whether the models' ability to detect pain without texture information depends solely on the mesh geometry.
- **Single Synthetic Head (Front):**
  The dataset ”Meshes with Texture in Frontal View” is used to assess texture's effect on pain recognition model performance compared to meshes without texture and the base case.
- **Single Synthetic Head (Side):**
  This case is analyzed with the help of the dataset ”Meshes with Texture in Side View” to determine whether different viewing angles significantly affect the model.
- **Random Synthetic Head:**
  Using the ”Random Assignment of Synthetic Heads to Videos” dataset, this case explores synthetic data's collective impact on pain recognition accuracy with random human textures.
- **Synthetic Head (Multiple Perspectives):**
  The case explores the benefits of leveraging multiple perspectives using the ”Meshes with Textures in Front and Side View” dataset. The goal is to evaluate whether incorporating diverse visual information can improve the effectiveness of pain recognition models.

6.1.2 Effect of Video Quantity

- **One Patient with One Texture:**
  By using the ”Replacement each Test-Patient with One Synthetic Head Texture” dataset, this case assesses the effect of replacing one Test-Patient with one synthetic identity on the performance of the pain recognition model.
- **One Patient with Multiple Textures (2, 3, 5, 10):**
  This analysis aims to determine if model performance improves with increased synthetic heads per Test-Patient compared to using only real data, leveraging the ”Replacement each Test-Patient with Multiple Synthetic Head Textures (2, 3, 5, 10)” datasets.
- **Synthetic Data + Real Data:**
  Utilizing the ”Synthetic Data + Real Data” dataset determines if combining synthetic and real data enhances model performance compared to using real data alone. By integrating synthetic data with real counterparts, the aim is to assess whether the combined dataset outperforms real data exclusively.

6.2 Evaluation Setup

The AUROC score measures recognition ability to evaluate the network's performance, indicating how well the model distinguishes between different classes. The epoch with the highest AUROC score on the synthetic data is used to assess the model's performance on the real human validation data.

In addition to the AUROC score, metrics such as the confusion matrix (to assess detection distribution), F1-score, precision, recall, and accuracy are collected to evaluate the model's overall performance comprehensively. These metrics offer insights into the model's effectiveness, such as its ability to classify positive and negative cases correctly and its overall accuracy in pain recognition tasks.

7 Results

7.1 Effect of Video Quality

Research Case	AUROC	F1-Score	Precision	Recall	Accuracy
Base Research Case BioVid Heat	0.741	0.666	0.644	0.692	0.654

Table 7.1: Training result of the **Base Research Case BioVid Heat** with **BioVid Heat Pain data as validation set** on the epoch with the highest AUROC-Score

Research Case	AUROC	F1-Score	Precision	Recall	Accuracy
Only Mesh	0.624	0.799	0.665	0.999	0.665
Single Synthetic Head (Front)	0.641	0.786	0.674	0.944	0.658
Single Synthetic Head (Side)	0.641	0.786	0.674	0.945	0.658
Random Synthetic Head	0.624	0.797	0.670	0.985	0.666
Synthetic Head (Multiple Perspectives)	0.653	0.799	0.667	0.997	0.666

Table 7.2: Training result with **synthetic data as validation set** on the epoch with the highest AUROC-Score

Table 7.1 presents the results of the base case evaluation of the trained baseline model, offering a detailed perspective on its performance across key metrics. The AUROC score of 0.741 indicates that the model can effectively discriminate between positive (pain) and negative (no pain) cases with relatively high confidence. The F1 score of 0.666 indicates a moderate performance level in precision and recall, suggesting a balance between correctly identifying pain expressions and avoiding false positives. With a precision score of 0.644, the model demonstrates a moderate level of accuracy in correctly identifying pain expressions, suggesting room for improvement in minimizing false positives. In contrast, the recall score of 0.692 implies that the model is relatively effective at identifying pain expressions when they are present, which is a positive indication of its ability to detect them. The accuracy score of 0.654 suggests that the model's overall performance is moderate, indicating the potential for further optimization to increase the proportion of correct predictions and reduce false positives. Notably, the precision score of 0.644 indicates a moderate level of accuracy in correctly identifying pain expressions, highlighting a key area for improvement to minimize false positives.

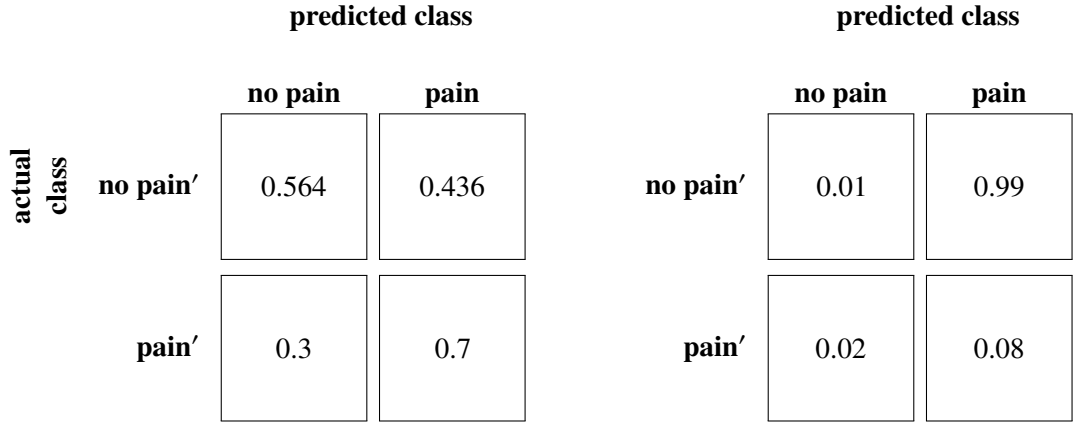
When meshes without texture in the frontal view were used, the model achieved an AUROC of 0.624, an F1 score of 0.799, a precision of 0.665, a recall of 0.999, and an accuracy of 0.665. This indicates a high recall rate, reassuring the audience about the model's adeptness at identifying

positive cases and highlighting the need for improvement in precision. For this case, the confusion matrix (7.3b) shows that the model correctly predicted a high proportion of actual positive cases but with an associated false positive rate of 0.99. The matrix highlights the model's high recall, as it successfully identifies nearly all actual positive cases, yet at the cost of high false positives. This reassures the audience about the model's adeptness at identifying positive cases, emphasizing its effectiveness in detecting pain expressions.

The model trained on meshes with texture in the frontal view produced slightly better AUROC (0.641), with an F1 score of 0.786, precision of 0.674, recall of 0.944, and accuracy of 0.658. This suggests an improvement in precision over the previous case. The confusion matrix (7.3c) indicates a more balanced performance with better class differentiation, instilling confidence in the model's reliability. This emphasis on the model's balanced performance enhances the audience's trust in its results.

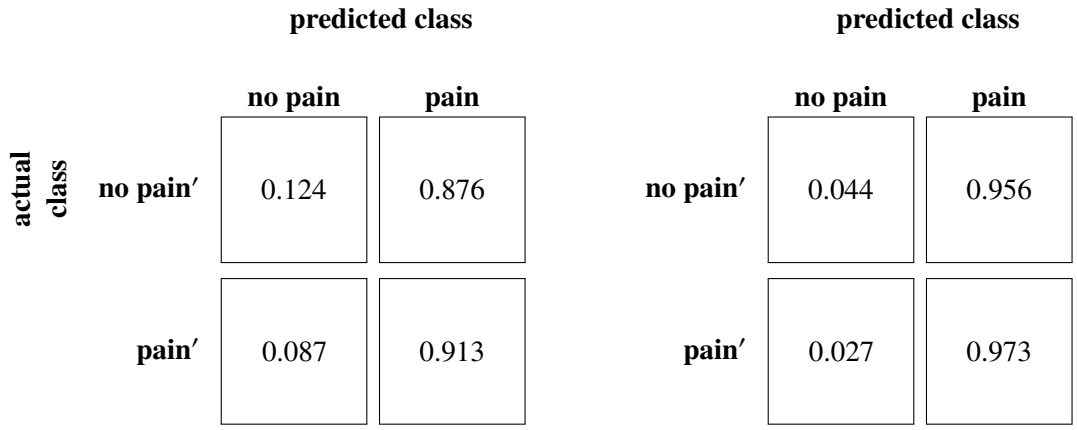
When meshes with texture and side view were employed, the model maintained the same AUROC of 0.641, F1 score of 0.786, and precision of 0.674 as the previous case. The recall rate slightly increased to 0.945, indicating better detection of positive cases. The confusion matrix (7.3f) reflects this improvement with a lower false negative rate.

In the random assignment of synthetic heads to videos, the model achieved an AUROC of 0.624, F1 score of 0.797, precision of 0.670, recall of 0.985, and accuracy of 0.666. This case shows high recall, suggesting efficient identification of positive cases, but precision could be improved. The confusion matrix (7.3e) shows the model's strong performance in correctly identifying true positives, yet improved precision is needed. In the case involving meshes with textures in front and side view, the model achieved the highest AUROC of 0.6533, an F1 score of 0.799, precision of 0.667, recall of 0.997, and accuracy of 0.666. This case demonstrated the model's ability to balance recall and precision, leading to a high overall performance and instilling confidence in its reliability.



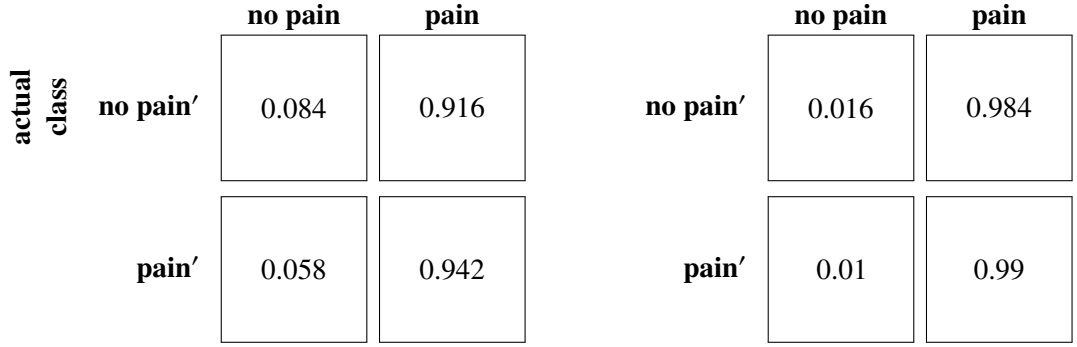
(a) Base Research Case BioVid Heat

(b) Only Mesh



(c) Single Synthetic Head (Front)

(d) Single Synthetic Head (Side)



(e) Random Synthetic Head

(f) Synthetic Head (Multiple Perspectives)

Table 7.3: Confusion Matrix of the individual cases with **synthetic data as validation set**

Research Case	AUROC	F1-Score	Precision	Recall	Accuracy
Only Mesh	0.551	0.8	0.666	1	0.666
Single Synthetic Head (Front)	0.543	0.793	0.668	0.977	0.661
Single Synthetic Head (Side)	0.559	0.798	0.668	0.922	0.666
Random Synthetic Head	0.57	0.739	0.684	0.805	0.623
Synthetic Head (Multiple Perspectives)	0.581	0.8	0.666	1	0.666

Table 7.4: Training result with **BioVid Heat Pain data as validation set** on the epoch with the highest AUROC-Score

In the case involving training on meshes without texture in a frontal view, the model achieved an AUROC score of 0.551, an F1 score of 0.8, a precision of 0.666, a recall of 1, and an accuracy of 0.666. The confusion matrix (7.5a) shows the model correctly identified 0.001 of true positives but predicted 0.999 false negatives, indicating a struggle to distinguish pain effectively.

When training on meshes with texture in a frontal view, the model produced an AUROC score of 0.543, an F1 score of 0.793, a precision of 0.668, a recall of 0.977, and an accuracy of 0.661. The confusion matrix (7.5b) reveals a slight improvement over the previous case, with the model correctly identifying 0.029 true positives and predicting 0.971 false negatives.

For training on meshes with texture and side view, the model reached an AUROC score of 0.559, an F1 score of 0.798, a precision of 0.668, a recall of 0.922, and an accuracy of 0.666. The confusion matrix (7.5c) indicates the model performed well, correctly identifying 0.014 true positives and predicting 0.986 false negatives.

When synthetic heads were randomly assigned to videos, the model achieved an AUROC score of 0.57, an F1 score of 0.739, a precision of 0.684, a recall of 0.805, and an accuracy of 0.623. The confusion matrix (7.5d) shows that the model correctly identified 0.256 true positives while predicting 0.744 false negatives.

Finally, in the case involving training on meshes with textures in front and side views, the model recorded an AUROC score of 0.581, an F1 score of 0.8, a precision of 0.666, a recall of 1, and an accuracy of 0.666. The confusion matrix (7.5e) reveals the model correctly identified 0.175 true positives while predicting 0.825 false negatives.

		predicted class: no pain	predicted class: pain
actual class	no pain′	0.001	0.999
	pain′	0.001	0.999

(a) Research Case Only Mesh

		predicted class: no pain	predicted class: pain
	no pain′	0.029	0.971
	pain′	0.023	0.977

(b) Research Case Single Synthetic Head (Front)

		predicted class: no pain	predicted class: pain
actual class	no pain′	0.014	0.986
	pain′	0.008	0.992

(c) Research Case Single Synthetic Head (Side)

		predicted class: no pain	predicted class: pain
	no pain′	0.256	0.744
	pain′	0.195	0.805

(d) Random Synthetic Head

	predicted class: no pain	predicted class: pain
no pain′	0.175	0.825
pain′	0.111	0.889

(e) Single Synthetic Head (Multiple Perspectives)

Table 7.5: Confusion Matrix results with **BioVid Heat Pain as validation set** on the epoch with the highest AUROC-Score

7.2 Effect of Video Quantity

Research Case	AUROC	F1-Score	Precision	Recall	Accuracy
1 Patient with 1 Texture	0.643	0.776	0.664	0.935	0.667
1 Patient with 2 Textures	0.629	0.752	0.691	0.816	0.634
1 Patient with 3 Textures	0.637	0.745	0.674	0.926	0.647
1 Patient with 5 Textures	0.642	0.767	0.682	0.910	0.664
1 Patient with 10 Textures	0.655	0.799	0.67	0.988	0.668
Synthetic Data + Real Data	0.632	0.774	0.684	0.896	0.654

Table 7.6: Training result with **synthetic data as validation set** on the epoch with the highest AUROC-Score

In the first case, where each test patient was replaced with one synthetic head texture, the confusion matrix (7.7a) reveals that the model correctly classified 0.122 true negatives and 0.915 true positives. The model mistakenly classified 0.878 false negatives and 0.085 false positives. The case indicates that the model may struggle with false negatives, suggesting that further improvements in classification could be beneficial.

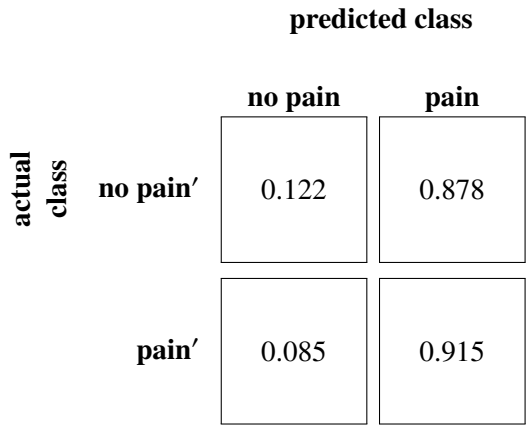
In the second case, where each test patient was replaced with multiple synthetic head textures (2 textures), the confusion matrix (7.7b) shows 0.185 true negatives and 0.87 true positives. The model classified 0.815 false negatives and 0.13 false positives. Although the recall is high, the model may benefit from reducing false negatives and improving precision.

The case involving multiple synthetic head textures (3 textures) achieves true negatives of 0.034 and true positives of 0.98, while false negatives and false positives are 0.966 and 0.02, respectively. This confusion matrix (7.7c) suggests better class differentiation, indicating the model's improved performance.

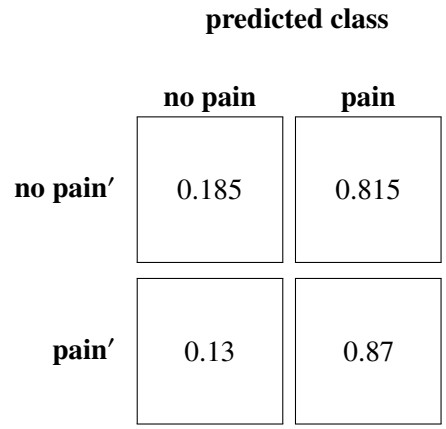
In the case with five synthetic head textures, the confusion matrix (7.7d) indicates that the model correctly classified 0.187 of true negatives and 0.874 of true positives, with false negatives and false positives at 0.813 and 0.126, respectively. This case shows an improvement in the balance of performance, suggesting potential gains in training models with increased diversity.

In the final case, with ten synthetic head textures, the confusion matrix (7.7e) results demonstrate true negatives at 0.023 and true positives at 0.99. False negatives and false positives are 0.977 and 0.01, respectively. These results show excellent performance in identifying true positives, but the model may require further adjustments to minimize false negatives.

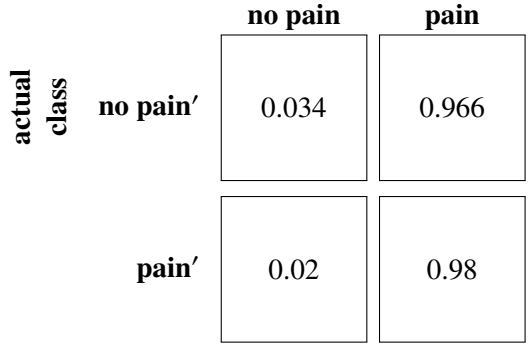
The final case involving the combination of synthetic and real data exhibits a confusion matrix (7.7f) of 0.175 true negatives and 0.889 true positives, with false negatives and false positives of 0.825 and 0.111, respectively. This case suggests a balance in performance across the metrics, potentially indicating the benefit of combining synthetic and real data in the training process.



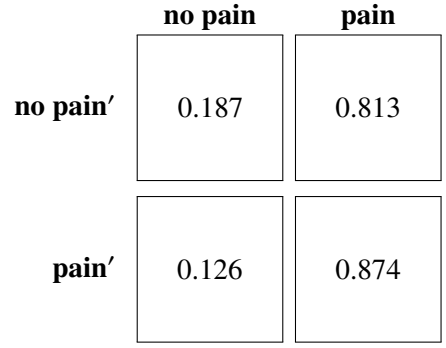
(a) 1 Patient with 1 Texture



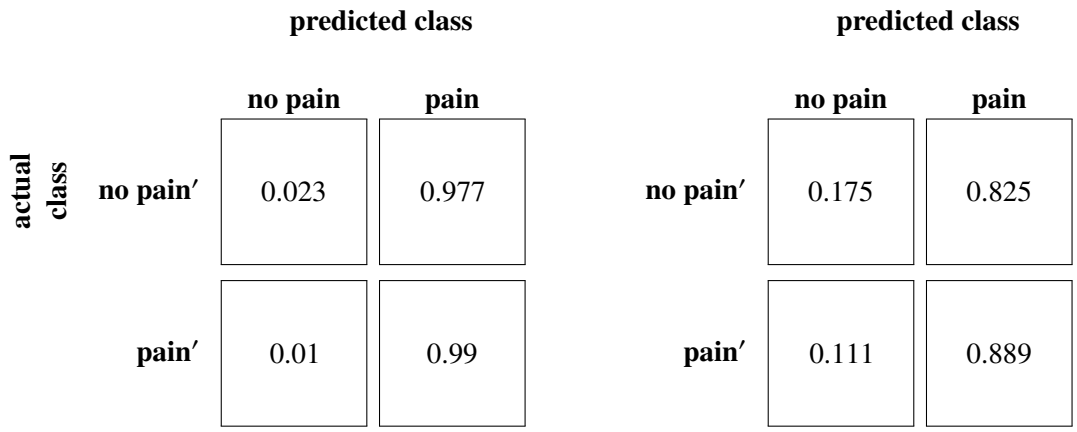
(b) 1 Patient with 2 Textures



(c) 1 Patient with 3 Textures



(d) 1 Patient with 5 Textures



(e) 1 Patient with 10 Textures

(f) Synthetic Data + Real Data

Table 7.7: Confusion Matrix results with **synthetic data as validation set** on the epoch with the highest AUROC-Score

Research Case	AUROC	F1-Score	Precision	Recall	Accuracy
1 Patient with 1 Texture	0.543	0.735	0.675	0.804	0.603
1 Patient with 2 Textures	0.56	0.724	0.682	0.776	0.608
1 Patient with 3 Textures	0.553	0.769	0.674	0.789	0.643
1 Patient with 5 Textures	0.57	0.792	0.668	0.973	0.66
1 Patient with 10 Textures	0.577	0.798	0.667	0.993	0.665
Synthetic Data + Real Data	0.78	0.817	0.7	0.981	0.708

Table 7.8: Training result with **BioVid Heat Pain data as validation set** on the epoch with the highest AUROC-Score

When each test patient is replaced with one synthetic head texture, the model demonstrates a confusion matrix (7.9a) in which it correctly classifies 0.564 true negatives and 0.3 true positives, indicating that this approach may struggle to correctly identify positive cases.

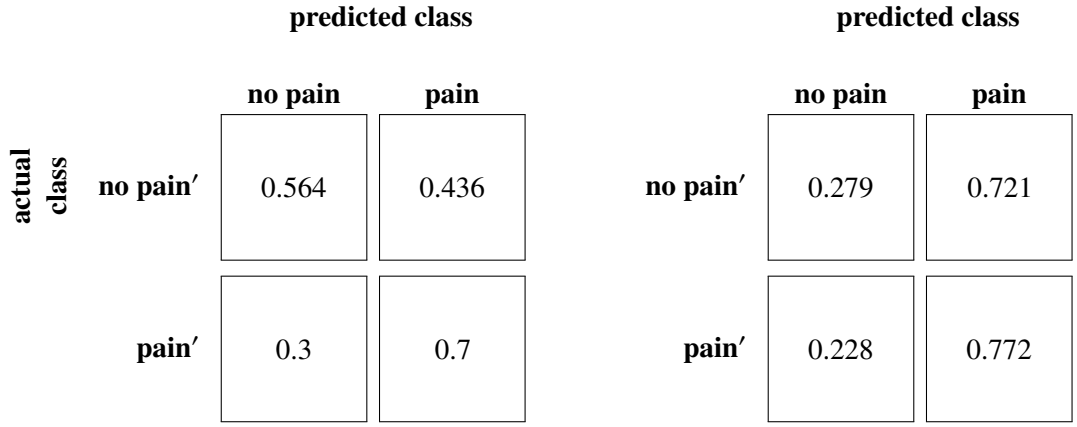
When each test patient is replaced with two synthetic head textures, the model improves performance, correctly identifying 0.279 true negatives and 0.228 true positives. This case demonstrates an overall enhancement in classification capabilities.

In the case with three synthetic head textures per test patient, the model's confusion matrix (7.9c) indicates a higher true negative rate of 0.145 and a true positive rate of 0.176, suggesting an increased ability to classify positive cases compared to the previous cases.

When five synthetic head textures are used per test patient, the model significantly improves by correctly classifying 0.034 true negatives and 0.027 true positives. This case reveals an enhanced capability to identify positive cases correctly.

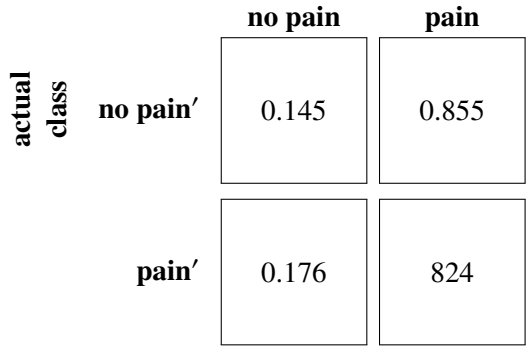
The case with ten synthetic head textures per test patient presents a balanced confusion matrix (7.9e), with the model correctly classifying 0.516 true negatives and 0.499 true positives. This case exhibits a more balanced approach in classifying negative and positive cases.

Finally, combining synthetic and real data, the case achieves the best performance across all metrics. In this case, the confusion matrix (7.9f) demonstrates a notable improvement in the model's classification capabilities, correctly identifying 0.162 true negatives and 0.094 true positives. This case's results indicate that leveraging real and synthetic data improves overall performance.

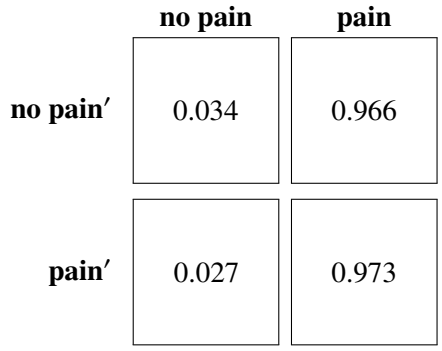


(a) 1 Patient with 1 Texture

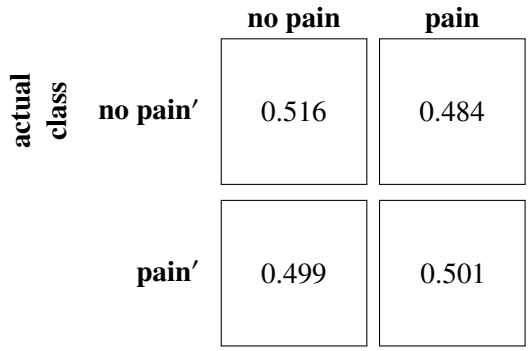
(b) 1 Patient with 2 Textures



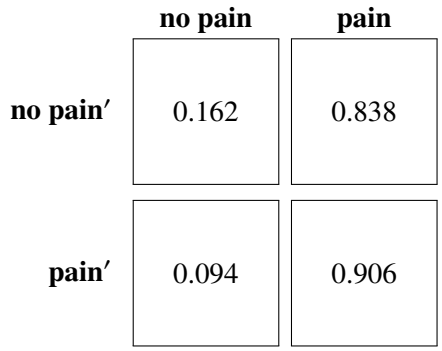
(c) 1 Patient with 3 Textures



(d) 1 Patient with 5 Textures



(e) 1 Patient with 10 Textures



(f) Synthetic Data + Real Data

Table 7.9: Confusion Matrix results with **BioVid Heat Pain as validation set** on the epoch with the highest AUROC-Score

8 Conclusion and Outlook

This thesis investigated the use of synthetic data generated through deep learning techniques to recognize pain in humans. It introduced machine learning, focusing on deep learning and convolutional neural networks (CNNs). Additionally, it discussed the importance of metrics for model evaluation and 3D modeling, specifically mesh creation for human heads.

Using the BioVid Heat Pain Database, synthetic video data replicating the facial expressions in the original videos was generated. The EMOCA repository provided the meshes, the FFHQ-UV repository provided the textures, and Blender facilitated the rendering of these synthetic videos. This process allowed for creating a diverse dataset with variations in human textures and recording angles. Subsequently, a SlowFast network was trained on different dataset combinations to explore the effectiveness of synthetic data-based models in pain recognition from real humans.

The results show that relying solely on synthetic data has certain performance limitations. However, incorporating synthetic and real data leads to improved pain recognition capabilities. This combined approach can leverage the strengths of both real and synthetic datasets, resulting in a more robust and effective model for pain recognition.

This research opens doors for exciting future directions in pain recognition using synthetic data. Potential next steps that can build upon this work could be:

1. **Expanding Dataset Diversity:**
   While diverse regarding facial textures, the current dataset could benefit from further expansion to enhance model generalizability. This includes:
   - Incorporate new perspectives and different lightning exposure for a more diverse dataset.
   - Simulating a broader spectrum of pain expressions.
   - Introducing environmental variations.
2. **Leveraging Generative AI for Dataset Diversification:**
   Generative AI techniques, such as Stable Diffusion [93], can potentially expand the scope of the synthetic dataset significantly. Stable Diffusion can generate new and unique facial textures, expressions, and lighting conditions that can be incorporated into the training data.
3. **Utilizing Additional Pain Recognition Datasets:**
   In addition to the current dataset, investigating other publicly available pain recognition datasets may prove advantageous. Such datasets include the UNBC-McMaster Shoulder Pain Expression Archive Database [23], which captures facial expressions of individuals experiencing shoulder pain during range-of-motion tests.

4. **Exploring Alternative Pain Recognition Models:**
   The current research focused on a binary pain classification (pain vs. no pain). Future studies can explore models that predict pain on a more granular scale. The five-step model (BL, PA1-PA4) outlined by Werner et al. [81] could be adapted for synthetic data-based pain recognition.
5. **Integrating Biomedical Signals:**
   Beyond facial expressions, pain recognition can be enhanced by incorporating additional data sources such as biomedical signals. These might include electrodermal activity (EDA), electrocardiogram (ECG), and electromyography (EMG).

Bibliography

[1] R. Danecek, M. J. Black, T. Bolkart. "EMOCA: Emotion Driven Monocular Face Capture and Animation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 20311–20322 (cit. on pp. 3, 5, 42, 59).

[2] P. P. Filntisis, G. Retsinas, F. Paraperas-Papantoniou, A. Katsamanis, A. Roussos, P. Maragos. "Visual Speech-Aware Perceptual 3D Facial Expression Reconstruction from Videos". In: *arXiv preprint arXiv:2207.11094* (2022) (cit. on pp. 3, 5, 42).

[3] H. Zhu, W. Wu, W. Zhu, L. Jiang, S. Tang, L. Zhang, Z. Liu, C. C. Loy. "CelebV-HQ: A Large-Scale Video Facial Attributes Dataset". In: *ECCV*. 2022 (cit. on pp. 3, 5, 51).

[4] H. Bai, D. Kang, H. Zhang, J. Pan, L. Bao. "FFHQ-UV: Normalized Facial UV-Texture Dataset for 3D Face Reconstruction". In: *IEEE Conference on Computer Vision and Pattern Recognition*. June 2023 (cit. on pp. 3, 5, 43–45, 47, 50).

[5] *Ensuring ethical standards and procedures for research with human beings — who.int*. https://www.who.int/activities/ensuring-ethical-standards-and-procedures-for-research-with-human-beings. [Accessed 25-04-2024] (cit. on p. 15).

[6] M. X. Patel, V. Doku, L. Tennakoon. "Challenges in recruitment of research participants". In: *Advances in Psychiatric Treatment* 9.3 (2003), pp. 229–238. DOI: 10.1192/apt.9.3.229 (cit. on p. 15).

[7] L. Zhou, S. Pan, J. Wang, A. V. Vasilakos. "Machine learning on big data: Opportunities and challenges". In: *Neurocomputing* 237 (2017), pp. 350–361. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.01.026. URL: https://www.sciencedirect.com/science/article/pii/S0925231217300577 (cit. on p. 15).

[8] *Terminology | International Association for the Study of Pain — iasp-pain.org*. https://www.iasp-pain.org/resources/terminology/#pain. [Accessed 07-04-2024] (cit. on p. 17).

[9] J. Thevenot, M. Bordallo Lopez, A. Hadid. "A Survey on Computer Vision for Assistive Medical Diagnosis From Faces". In: *IEEE Journal of Biomedical and Health Informatics* PP (Oct. 2017), pp. 1–1. DOI: 10.1109/JBHI.2017.2754861 (cit. on p. 17).

[10] R. S. G. Dennis C. Turk, M. E. Bombardier. *Clinical Pain Measurement for Physical Therapists*. Jones Bartlett Learning, 2017. URL: https://www.ncbi.nlm.nih.gov/books/NBK556098/ (cit. on p. 17).

[11] M. Diana J. Moritz M.D. "Assessment of Pain in Patients With Dementia". In: *Journal of the American Geriatrics Society* 65.5 (2017), pp. 940–948. DOI: 10.1111/jgs.14823. URL: https://pubmed.ncbi.nlm.nih.gov/37293681/ (cit. on p. 17).

[12] K. W. Prkachin. "The utility of facial expression in pain assessment". In: *Pain* 51.1 (1992), pp. 197–210. DOI: 10.1058/913992. URL: https://www.sciencedirect.com/science/article/pii/105891399290001S (cit. on p. 17).

[13] T. C. Dildine, C. M. Amir, J. Parsons, L. Y. Atlas. "How Pain-Related Facial Expressions Are Evaluated in Relation to Gender, Race, and Emotion". In: *Affective Science* 4.2 (Mar. 2023), pp. 350–369. ISSN: 2662-205X. DOI: 10.1007/s42761-023-00181-6. URL: http://dx.doi.org/10.1007/s42761-023-00181-6 (cit. on p. 17).

[14] *Facial Action Coding System — paulekman.com*. https://www.paulekman.com/facial-action-coding-system/. [Accessed 07-04-2024] (cit. on p. 17).

[15] C. Correia-Caeiro, A. Burrows, D. A. Wilson, A. Abdelrahman, T. Miyabe-Nishiwaki. "CalliFACS: The common marmoset Facial Action Coding System". In: *PLOS ONE* 17.5 (May 2022). Ed. by K. E. Slocombe, e0266442. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0266442. URL: http://dx.doi.org/10.1371/journal.pone.0266442 (cit. on p. 17).

[16] M. Tavakolian, A. Hadid. "A Spatiotemporal Convolutional Neural Network for Automatic Pain Intensity Estimation from Facial Dynamics". In: *International Journal of Computer Vision* 127.10 (June 2019), pp. 1413–1425. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01191-3. URL: http://dx.doi.org/10.1007/s11263-019-01191-3 (cit. on p. 18).

[17] P. Ekman, W. V. Friesen. "Facial action coding system". In: *Environmental Psychology & Nonverbal Behavior* (1978) (cit. on pp. 18, 19).

[18] R. Zhi, M. Liu, D. Zhang. "A comprehensive survey on automatic facial action unit analysis". In: *The Visual Computer* 36 (May 2020). DOI: 10.1007/s00371-019-01707-5 (cit. on pp. 18, 19).

[19] J. Cohn. "Foundations of Human Computing: Facial Expression and Emotion". In: Jan. 2006, pp. 233–238. ISBN: 978-3-540-72346-2. DOI: 10.1007/978-3-540-72348-6_1 (cit. on p. 19).

[20] J. Harrigan, R. Rosenthal, K. Scherer. *New handbook of methods in nonverbal behavior research*. Oxford University Press, 2008 (cit. on p. 19).

[21] Z. Hammal, J. F. Cohn. "Automatic detection of pain intensity". In: *Proceedings of the 14th ACM International Conference on Multimodal Interaction*. ICMI '12. Santa Monica, California, USA: Association for Computing Machinery, 2012, pp. 47–52. ISBN: 9781450314671. DOI: 10.1145/2388676.2388688. URL: https://doi.org/10.1145/2388676.2388688 (cit. on pp. 19, 20).

[22] K. M. Prkachin, P. E. Solomon. "The structure, reliability and validity of pain expression: Evidence from patients with shoulder pain". In: *PAIN* 139.2 (2008), pp. 267–274. ISSN: 0304-3959. DOI: https://doi.org/10.1016/j.pain.2008.04.010. URL: https://www.sciencedirect.com/science/article/pii/S0304395908001991 (cit. on pp. 19, 20).

[23] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, I. Matthews. "Painful data: The UNBC-McMaster shoulder pain expression archive database". In: *2011 IEEE International Conference on Automatic Face Gesture Recognition (FG)*. 2011, pp. 57–64. DOI: 10.1109/FG.2011.5771462 (cit. on pp. 20, 75).

[24] S. Walter, S. Gruss, H. Ehleiter, J. Tan, H. C. Traue, P. Werner, A. Al-Hamadi, S. Crawcour, A. O. Andrade, G. Moreira da Silva. "The biovid heat pain database data for the advancement and systematic validation of an automated pain recognition system". In: *2013 IEEE International Conference on Cybernetics (CYBCO)*. 2013, pp. 128–131. DOI: 10.1109/CYBConf.2013.6617456 (cit. on pp. 20, 39, 47).

[25] M. Mohammed, M. Khan, E. Bashier. *Machine Learning: Algorithms and Applications*. July 2016. ISBN: 9781498705387. DOI: 10.1201/9781315371658 (cit. on pp. 20–22).

[26] T. M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2 (cit. on p. 20).

[27] S. Shalev-Shwartz, S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014, pp. I–XVI, 1–397. ISBN: 978-1-10-705713-5 (cit. on p. 21).

[28] A. Chouchane, A. Ouamane, Y. Himeur, W. Mansoor, S. Atalla, A. Benzaibak, C. Boudellal. *Improving CNN-based Person Re-identification using score Normalization*. 2023. DOI: 10.48550/ARXIV.2307.00397. URL: https://arxiv.org/abs/2307.00397 (cit. on p. 21).

[29] E. Núñez, E. Steyerberg, J. Nuñez Villota. "Regression Modeling Strategies". In: *Revista española de cardiología* 64 (June 2011), pp. 501–7. DOI: 10.1016/j.rec.2011.01.017 (cit. on p. 21).

[30] Y. Huang, P. Mccullagh, N. Black, R. Harper. "Feature Selection and Classification Model Construction on Type 2 Diabetic Patient's Data". In: vol. 3275. Nov. 2004, pp. 349–364. ISBN: 978-3-540-24054-9. DOI: 10.1007/978-3-540-30185-1_17 (cit. on p. 21).

[31] J. Mueller, L. Massaron. *Machine Learning For Dummies*. For dummies. Wiley, 2016. ISBN: 9781119245513. URL: https://books.google.de/books?id=JLEyDAAAQBAJ (cit. on p. 22).

[32] V. N. Ogar, S. Hussain, K. A. Gamage. "The use of artificial neural network for low latency of fault detection and localisation in transmission line". In: *Heliyon* 9.2 (Feb. 2023), e13376. ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2023.e13376. URL: http://dx.doi.org/10.1016/j.heliyon.2023.e13376 (cit. on p. 22).

[33] I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, S. F. dos Reis Alves. "Artificial Neural Network Architectures and Training Processes". In: *Artificial Neural Networks : A Practical Course*. Cham: Springer International Publishing, 2017, pp. 21–28. ISBN: 978-3-319-43162-8. DOI: 10.1007/978-3-319-43162-8_2. URL: https://doi.org/10.1007/978-3-319-43162-8_2 (cit. on p. 22).

[34] https://www.surfactants.net/2022/09/. [Accessed 28-04-2024]. 2022 (cit. on p. 22).

[35] F. Bre, J. Gimenez, V. Fachinotti. "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". In: *Energy and Buildings* 158 (Nov. 2017). DOI: 10.1016/j.enbuild.2017.11.045 (cit. on p. 23).

[36] g. anuradhac arvindpdmn gurumoorthyP arvindpdmn. *Artificial Neural Network — devopedia.org*. https://devopedia.org/artificial-neural-network. [Accessed 06-04-2024] (cit. on p. 23).

[37] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 23–25).

[38] R. Pramoditha. *Overview of a Neural Network's Learning Process — medium.com*. https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61690a502fa. [Accessed 19-04-2024] (cit. on p. 24).

[39] S. Haykin. *Neural Networks and Learning Machines*. Pearson International Edition. Pearson, 2009. ISBN: 9780131293762. URL: https://books.google.de/books?id=KCwWOAAACAAJ (cit. on p. 24).

[40] D. E. Rumelhart, G. E. Hinton, R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: http://dx.doi.org/10.1038/323533a0 (cit. on p. 24).

[41] M. Mishra. *Convolutional Neural Networks, Explained — towardsdatascience.com*. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939. [Accessed 07-04-2024] (cit. on p. 25).

[42] *ML Practicum: Image Classification | Machine Learning | Google for Developers — developers.google.com*. https://developers.google.com/machine-learning/practica/image-classification/. [Accessed 07-04-2024] (cit. on pp. 25–28).

[43] Y. Guo, Y. Liu, A. Oerlemans, S.-Y. Lao, S. Wu, M. Lew. "Deep learning for visual understanding: A review". In: *Neurocomputing* 187 (Nov. 2015). DOI: 10.1016/j.neucom.2015.09.116 (cit. on pp. 25–28).

[44] W. Rawat, Z. Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review". In: *Neural Computation* 29 (June 2017), pp. 1–98. DOI: 10.1162/NECO_a_00990 (cit. on pp. 26, 27).

[45] D. Lewenko. *Image Classification with Deep Learning: A theoretical introduction to machine learning and deep... — medium.com*. https://medium.com/analytics-vidhya/image-classification-with-deep-learning-a-theoretical-introduction-to-machine-learning-and-deep-d118905c6d3a. [Accessed 28-04-2024] (cit. on p. 26).

[46] A. Krizhevsky, I. Sutskever, G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 28).

[47] H. Allamy. "METHODS TO AVOID OVER-FITTING AND UNDER-FITTING IN SUPERVISED MACHINE LEARNING (COMPARATIVE STUDY)". In: (Dec. 2014) (cit. on p. 28).

[48] A. Halevy, P. Norvig, F. Pereira. "The Unreasonable Effectiveness of Data". In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12. DOI: 10.1109/MIS.2009.36 (cit. on p. 29).

[49] L. Perez, J. Wang. *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. 2017. DOI: 10.48550/ARXIV.1712.04621. URL: https://arxiv.org/abs/1712.04621 (cit. on p. 29).

[50] C. Shorten, T. M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (July 2019). ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: http://dx.doi.org/10.1186/s40537-019-0197-0 (cit. on p. 29).

[51] J. Brownlee. *What is the Difference Between Test and Validation Datasets? - MachineLearningMastery.com — machinelearningmastery.com*. https://machinelearningmastery.com/difference-test-validation-datasets/. [Accessed 07-04-2024] (cit. on p. 29).

[52] *Confusion Matrix in Machine Learning - GeeksforGeeks — geeksforgeeks.org*. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/. [Accessed 07-04-2024] (cit. on p. 30).

[53] *What is a confusion matrix? | IBM — ibm.com*. https://www.ibm.com/topics/confusion-matrix. [Accessed 07-04-2024] (cit. on pp. 30, 31).

[54] *Evaluation measures for multiclass problems — gabrielelanaro.github.io*. http://gabrielelanaro.github.io/blog/2016/02/03/multiclass-evaluation-measures.html. [Accessed 07-04-2024] (cit. on p. 31).

[55] *Classification: Accuracy | Machine Learning | Google for Developers — developers.google.com*. https://developers.google.com/machine-learning/crash-course/classification/accuracy. [Accessed 07-04-2024] (cit. on p. 32).

[56] *Accuracy &x2014; PyTorch-Metrics 1.3.2 documentation — lightning.ai*. https://lightning.ai/docs/torchmetrics/stable/classification/accuracy.html. [Accessed 07-04-2024] (cit. on p. 32).

[57] N. Bressler. *How to Check the Accuracy of Your Machine Learning Model | Deepchecks — deepchecks.com*. https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/. [Accessed 07-04-2024] (cit. on p. 32).

[58] *Accuracy vs. precision vs. recall in machine learning: what's the difference? — evidentlyai.com*. https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall. [Accessed 07-04-2024] (cit. on p. 32).

[59] P. Flach, M. Kull. "Precision-Recall-Gain Curves: PR Analysis Done Right". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/33e8075e9970de0cfea955afd4644bb2-Paper.pdf (cit. on p. 33).

[60] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738 (cit. on p. 33).

[61] H. He, E. A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239 (cit. on pp. 33, 34).

[62] *Evaluation Metrics, ROC-Curves and imbalanced datasets — davidsbatista.net*. https://www.davidsbatista.net/blog/2018/08/19/NLP_Metrics/. [Accessed 07-04-2024] (cit. on p. 33).

[63] G. Forman. "An extensive empirical study of feature selection metrics for text classification [J]". In: *Journal of Machine Learning Research - JMLR* 3 (Mar. 2003) (cit. on p. 33).

[64] *F-1 Score &x2014; PyTorch-Metrics 1.3.2 documentation — lightning.ai*. https://lightning.ai/docs/torchmetrics/stable/classification/f1_score.html. [Accessed 07-04-2024] (cit. on p. 34).

[65] J. Plowman. *3D game design with unreal engine 4 and blender*. Birmingham, England: Packt Publishing, Jan. 2016 (cit. on pp. 35, 36).

[66] J. Patnode. *Character Modeling with Maya and Zbrush: Professional Polygonal Modeling Techniques*. Elsevier/Focal, 2008. ISBN: 9780240520346. URL: https://books.google.de/books?id=gqn4FIlYpZwC (cit. on pp. 35, 36).

[67] *Polygon mesh - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Polygon_mesh. [Accessed 07-04-2024] (cit. on p. 35).

[68] R. Caudron, P. Nicq. *Blender 3D By Example*. Packt Publishing, 2015. ISBN: 9781785280245. URL: https://books.google.de/books?id=gtdOCwAAQBAJ (cit. on p. 37).

[69] D. L. Martinez, O. Rudovic, R. Picard. *Personalized Automatic Estimation of Self-reported Pain Intensity from Facial Expressions*. 2017. DOI: 10.48550/ARXIV.1706.07154. URL: https://arxiv.org/abs/1706.07154 (cit. on p. 39).

[70] M. A. Haque, R. B. Bautista, F. Noroozi, K. Kulkarni, C. B. Laursen, R. Irani, M. Bellantonio, S. Escalera, G. Anbarjafari, K. Nasrollahi, O. K. Andersen, E. G. Spaich, T. B. Moeslund. "Deep Multimodal Pain Recognition: A Database and Comparison of Spatio-Temporal Visual Modalities". In: *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*. 2018, pp. 250–257. DOI: 10.1109/FG.2018.00044 (cit. on p. 39).

[71] P. Rodriguez, G. Cucurull, J. Gonzàlez, J. M. Gonfaus, K. Nasrollahi, T. B. Moeslund, F. X. Roca. "Deep Pain: Exploiting Long Short-Term Memory Networks for Facial Expression Classification". In: *IEEE Transactions on Cybernetics* 52.5 (2022), pp. 3314–3324. DOI: 10.1109/TCYB.2017.2662199 (cit. on p. 39).

[72] F. Wang, X. Xiang, C. Liu, T. D. Tran, A. Reiter, G. D. Hager, H. Quon, J. Cheng, A. L. Yuille. "Regularizing face verification nets for pain intensity regression". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 1087–1091. DOI: 10.1109/ICIP.2017.8296449 (cit. on p. 39).

[73] S. R. Richter, V. Vineet, S. Roth, V. Koltun. *Playing for Data: Ground Truth from Computer Games*. 2016. DOI: 10.48550/ARXIV.1608.02192. URL: https://arxiv.org/abs/1608.02192 (cit. on p. 40).

[74] S. R. Richter, Z. Hayder, V. Koltun. *Playing for Benchmarks*. 2017. DOI: 10.48550/ARXIV.1709.07322. URL: https://arxiv.org/abs/1709.07322 (cit. on p. 40).

[75] P. Krahenbuhl. "Free Supervision from Video Games". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2955–2964. DOI: 10.1109/CVPR.2018.00312 (cit. on p. 40).

[76] A. Roitberg, D. Schneider, A. Djamal, C. Seibold, S. Reiß, R. Stiefelhagen. *Let's Play for Action: Recognizing Activities of Daily Living by Learning from Life Simulation Video Games*. 2021. DOI: 10.48550/ARXIV.2107.05617. URL: https://arxiv.org/abs/2107.05617 (cit. on p. 40).

[77] K. Pikulkaew, V. Chouvatut. "Enhanced Pain Detection and Movement of Motion with Data Augmentation based on Deep Learning". In: *2021 13th International Conference on Knowledge and Smart Technology (KST)*. 2021, pp. 197–201. DOI: 10.1109/KST51265.2021.9415827 (cit. on p. 40).

[78] T. Li, T. Bolkart, M. J. Black, H. Li, J. Romero. "Learning a model of facial shape and expression from 4D scans". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36.6 (2017), 194:1–194:17. URL: https://doi.org/10.1145/3130800.3130813 (cit. on p. 41).

[79] T. Karras, S. Laine, T. Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2018. DOI: 10.48550/ARXIV.1812.04948. URL: https://arxiv.org/abs/1812.04948 (cit. on p. 43).

[80] P. Werner, A. Al-Hamadi, S. Walter. "Analysis of Facial Expressiveness During Experimentally Induced Heat Pain". In: Oct. 2017. DOI: 10.1109/ACIIW.2017.8272610 (cit. on pp. 48, 49).

[81] P. Werner, A. Al-Hamadi, R. Niese, S. Walter, S. Gruss, H. Traue. "Towards Pain Monitoring: Facial Expression, Head Pose, a new Database, an Automatic System and Remaining Challenges". In: Sept. 2013. DOI: 10.5244/C.27.119 (cit. on pp. 49, 76).

[82] *GitHub - neverhood311/Stop-motion-OBJ: A Blender add-on for importing a sequence of OBJ meshes as frames — github.com*. https://github.com/neverhood311/Stop-motion-OBJ/tree/master. [Accessed 12-04-2024] (cit. on p. 51).

[83] T. Martyniuk, O. Kupyn, Y. Kurlyak, I. Krashenyi, J. Matas, V. Sharmanska. "DAD-3DHeads: A Large-scale Dense, Accurate and Diverse Dataset for 3D Head Alignment from a Single Image". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022 (cit. on p. 52).

[84] Y. Choi, Y. Uh, J. Yoo, J.-W. Ha. "StarGAN v2: Diverse Image Synthesis for Multiple Domains". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on p. 52).

[85] *PyTorch3D · A library for deep learning with 3D data — pytorch3d.org*. https://pytorch3d.org. [Accessed 15-04-2024] (cit. on p. 52).

[86] R. M. Al-Eidan, H. Al-Khalifa, A. Al-Salman. "Deep-Learning-Based Models for Pain Recognition: A Systematic Review". In: *Applied Sciences* 10.17 (2020). ISSN: 2076-3417. DOI: 10.3390/app10175984. URL: https://www.mdpi.com/2076-3417/10/17/5984 (cit. on p. 59).

[87] J. Carreira, A. Zisserman. *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*. 2017. DOI: 10.48550/ARXIV.1705.07750. URL: https://arxiv.org/abs/1705.07750 (cit. on p. 59).

[88] C. Feichtenhofer, H. Fan, J. Malik, K. He. "SlowFast Networks for Video Recognition". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6201–6210. DOI: 10.1109/ICCV.2019.00630 (cit. on pp. 59, 60).

[89] X. Wang, R. Girshick, A. Gupta, K. He. "Non-local Neural Networks". In: *arXiv: Computer Vision and Pattern Recognition* (2017) (cit. on p. 59).

[90] K. He, X. Zhang, S. Ren, J. Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on pp. 59, 60).

[91] S. Basodi, C. Ji, H. Zhang, Y. Pan. "Gradient amplification: An efficient way to train deep neural networks". In: *Big Data Mining and Analytics* 3.3 (2020), pp. 196–207. DOI: 10.26599/BDMA.2020.9020004 (cit. on pp. 59, 60).

[92] S. Gkikas. *Biovid holdouteval (2023)*. https://www.nit.ovgu.de/nitmedia/Bilder/Dokumente/BIOVIDDokumente/BioVidHoldOutEvalProposal.pdf. [Accessed 17-04-2024] (cit. on p. 61).

[93] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV] (cit. on p. 75).

All links were last followed on April 30, 2024.